#2

11017 U.S. PTO
10/015604
12/17/01

Docket No. 217398US2RD

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Shinichiro HAMADA, et al. GAU:

SERIAL NO: NEW APPLICATION EXAMINER:

FILED: HEREWITH

FOR: METHOD AND APPARATUS FOR EDITING WEB DOCUMENT FROM PLURALITY OF WEB SITE INFORMATION

REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

SIR:

☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §120**.

☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §119(e)**.

☒ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
Japan	2000-383625	December 18, 2000

Certified copies of the corresponding Convention Application(s)

☒ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

☐ were filed in prior application Serial No. filed

☐ were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

☐ (B) Application Serial No.(s)

- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak
Registration No. 24,913

C. Irvin McClelland
Registration Number 21,124

22850
Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 10/98)

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２０００年１２月１８日

出願番号
Application Number: 特願２０００－３８３６２５

出願人
Applicant(s): 株式会社東芝

２００１年　８月３１日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３０７９８６６

【書類名】　特許願
【整理番号】　A000007604
【提出日】　平成12年12月18日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F　7/00
【発明の名称】　文書合成方法および文書合成装置
【請求項の数】　8
【発明者】
【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
【氏名】　浜田　伸一郎
【発明者】
【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
【氏名】　關　俊文
【特許出願人】
【識別番号】　000003078
【氏名又は名称】　株式会社　東芝
【代理人】
【識別番号】　100058479
【弁理士】
【氏名又は名称】　鈴江　武彦
【電話番号】　03-3502-3181
【選任した代理人】
【識別番号】　100084618
【弁理士】
【氏名又は名称】　村松　貞男
【選任した代理人】
【識別番号】　100068814

【弁理士】

【氏名又は名称】　坪井　淳

【選任した代理人】

【識別番号】　100092196

【弁理士】

【氏名又は名称】　橋本　良郎

【選任した代理人】

【識別番号】　100091351

【弁理士】

【氏名又は名称】　河野　哲

【選任した代理人】

【識別番号】　100088683

【弁理士】

【氏名又は名称】　中村　誠

【選任した代理人】

【識別番号】　100070437

【弁理士】

【氏名又は名称】　河井　将次

【手数料の表示】

【予納台帳番号】　011567

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　文書合成方法および文書合成装置

【特許請求の範囲】

【請求項１】　インターネットにおけるＷＷＷ（Ｗｏｒｌｄ　Ｗｉｄｅ　ｗｅｂ）上のマークアップ言語で記述された複数の第１の文書の内容の一部をＷＷＷ上のマークアップ言語で記述された第２の文書に合成するための文書合成方法であって、

少なくとも、前記第１の文書の該インターネット上の所在と、該第１の文書から抽出する部分文書の範囲と、前記第２の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第２の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第２の文書に従って、

前記第１の文書から前記部分文書を抽出して、その部分文書を前記第２の文書上の前記指定された合成位置に挿入するとともに、前記変換ルールを用いて前記第２の文書上の前記指定された範囲の文書構造を変換することで、前記第２の文書上に１または複数の前記部分文書を合成することを特徴とする文書合成方法。

【請求項２】　前記第２の文書は、少なくとも、前記第２の文書上の前記部分文書の挿入位置とを指定するとともに、前記第１の文書の所在と、該第１の文書から抽出する部分文書の範囲とを記述するため第１のタグと、

前記変換ルールを用いて文書構造を変換すべき範囲を指定するとともに、前記変換ルールを記述したファイルの識別情報を記述するための第２のタグと、

を用いて記述されていることを特徴とする請求項１記載の文書合成方法。

【請求項３】　前記第２の文書は、ＸＭＬ（Ｅｘｔｅｎｓｉｂｌｅ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ）で記述され、前記第１の文書がＸＭＬで記述されていないときは、まず、ＸＭＬによる記述型式に変換した後、前記第１の文書から前記部分文書を抽出して、その部分文書を前記第２の文書上の前記指定された挿入位置に挿入することを特徴とする請求項１記載の文書合成方法。

【請求項４】　インターネットにおけるＷＷＷ（Ｗｏｒｌｄ　Ｗｉｄｅ　ｗ

ｅｂ）上のマークアップ言語で記述された複数の第１の文書の内容の一部をＷＷＷ上のマークアップ言語で記述された第２の文書に合成する文書合成装置であって、

少なくとも、前記第１の文書の該インターネット上の所在と、該第１の文書から抽出する部分文書の範囲と、前記第２の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第２の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第２の文書に従って、前記第１の文書から前記部分文書を抽出して、その部分文書を前記第２の文書上の前記指定された挿入位置に挿入する挿入手段と、

前記第２の文書に従って、該第２の文書上の前記指定された範囲の文書構造を、前記変換ルールを用いて所望の文書構造に変換する変換手段と、

を具備し、

前記第２の文書上に１または複数の前記部分文書を合成することを特徴とする文書合成装置。

【請求項５】　前記第２の文書は、少なくとも、前記第２の文書上の前記部分文書の挿入位置とを指定するとともに、前記第１の文書の所在と、該第１の文書から抽出する部分文書の範囲とを記述するため第１のタグと、

前記変換ルールを用いて文書構造を変換すべき範囲を指定するとともに、前記変換ルールを記述したファイルの識別情報を記述するための第２のタグと、

を用いて記述されていることを特徴とする請求項４記載の文書合成装置。

【請求項６】　前記第２の文書は、ＸＭＬ（Ｅｘｔｅｎｓｉｂｌｅ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ）で記述されていることを特徴とする請求項４記載の文書合成装置。

【請求項７】　前記第１の文書がＸＭＬで記述されていないとき、該第１の文書をＸＭＬによる記述型式に変換する第２の変換手段をさらに具備し、

前記挿入手段は、ＸＭＬ文書の前記第１の文書から前記部分文書を抽出して、その部分文書を前記第２の文書上の前記指定された挿入位置に挿入することを特徴とする請求項４記載の文書合成装置。

【請求項８】　インターネットにおけるWWW（Ｗｏｒｌｄ　Ｗｉｄｅ　ｗｅｂ）上のマークアップ言語で記述された複数の第１の文書の内容の一部をマークアップ言語で記述された第２の文書に合成するための処理をコンピュータに実行させるためのプログラムであって、

少なくとも、前記第１の文書の該インターネット上の所在と、該第１の文書から抽出する部分文書の範囲と、前記第２の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第２の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第２の文書に従って、前記第１の文書から前記部分文書を抽出して、その部分文書を前記第２の文書上の前記指定された挿入位置に挿入するための処理と、

前記第２の文書に基づき、該第２の文書上の前記指定された範囲の文書構造を、前記変換ルールを用いて所望の文書構造に変換するための処理と、

をコンピュータに実行させるためのプログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、複数のウェブ文書を１つのウェブ文書上に合成するためのウェブ文書合成方法およびそれを用いたウェブ文書合成装置に関する。

【０００２】

【従来の技術】

WWW（Ｗｏｒｌｄ　Ｗｉｄｅ　Ｗｅｂ）は効果的なプレゼンテーションを低コストで構築・公開できる情報基盤として普及し、世界中のサイトで膨大な情報資源が公開されている。またWWWはサーバクライアントシステムのためのインフラの側面を持っている。特に電子商取引や最近ではＡＳＰ（Ａｐｐｌｉｃａｔｉｏｎ　Ｓｅｒｖｉｃｅ　Ｐｒｏｖｉｄｉｎｇ）などへの応用が期待されており、本格的なコマースサイトが急増しつつある状況にある。電子商取引では、ウェブページは、商取引を処理する企業内ＬＡＮのバックエンドシステムとユーザとを結ぶ操作パネルとしての役割を果たす。WWWはサイトを越えて世界中のコン

ピュータシステムをつなぐ唯一のインフラであるが、今後もウェブトップ指向への流れは続くことが予想される。

【０００３】

ＷＷＷで交換される情報資源は増加の一途をたどり、ウェブシステムに要求される処理はより複雑で多様なものになるだろう。

【０００４】

特に、企業はＷＷＷを積極的に活用しており、企業データやニュース・商品カタログ情報など自社の持つ大量のデータをウェブページを通じて公開しているが、各ウェブページを一から作るにはあまりにも人手がかかりすぎるため、定型的なコンテンツを含むウェブページについては、データベースから静的あるいは動的に機械生成する技術を導入しており、サイト構築および運用を効率化している。このようなウェブサイトの構築・運用ツールは、多くのソフトウェアベンダーから提供されており、非常に充実している。しかしこれらの技術はいずれも閉じた単一ウェブサイトの構築や運用の効率化・高性能化に関するものである。

【０００５】

単一ウェブサイトの構築・運用環境が整備された現在、次にＷＷＷに求められるのはウェブサイト間連携である。すなわちサーバクライアントシステムから分散システムへの発展である。特に本格的な電子商取引の時代を迎えるにあたり、各コマースサイトの電子商取引システムの連携は必須となる。

【０００６】

電子商取引システムの連携には、商品プロファイルなどのデータフォーマットや語彙の共通化、そして共通のビジネスモデル、それに従った共通のメッセージフォーマットやプロトコルなど多くの取り決めが必要である。これに対し、ＯＡＳＩＳやＢｉｚＴａｌｋなど業界団体が標準化を進めているが、企業間の利害の不一致や商習慣の違いなど多くの壁があるため、その成果が実を結ぶには、まだまだ時間を要することは間違いない。

【０００７】

一方でその火急のニーズに対応するため、各ソフトウェアベンダーからは、上述のウェブサイト構築・運用ツールにウェブサイトの連携機構を追加したパッケ

ージが提供されている。

【０００８】

しかし、データベースを中心に据えたアプリケーションロジック群を核とする従来的なシステム構築手法は、単一ウェブサイトに対してはウェブページを単なるユーザインターフェースとして位置付けることで有効に機能したが、複数ウェブサイトにまたがるシステムに対してはそのままでは適用できない。なぜなら、この構築手法ではシステム連携を実現するためにアプリケーションロジックを接続する必要があるが、サイト間はファイアウォールによってさえぎられており、ほとんどの場合ＨＴＴＰ以外のメッセージが交換できないからである。

【０００９】

従って、唯一のメッセージ交換のチャンネルであるＨＴＴＰをベースとしたシステム統合モデルが必要だが、パッケージの多くは従来のサイト構築技術にＨＴＴＰアクセス機能を追加しただけであり、ＨＴＴＰおよびＷＷＷの機能を生かしきれていない状況にある。

【００１０】

このようにサイト間のシステム連携は、それぞれのシステムが持つロジックを接続するために多くの取り決めが必要であり本質的に難しい課題である。

【００１１】

そこで、ロジック接続ではなくコンテンツ交換を用いたウェブサイト間連携を課題として着目してみると、ウェブサイト間コンテンツ連携は、ウェブリソースの構造変換程度の調節ですむため、ウェブサイト間システム連携に比べて解決すべき課題は少ない。

【００１２】

しかし、その一方で、コンテンツ連携がもたらす効果は十分に大きい。先に述べたようにＷＷＷではすでに膨大なウェブリソースが公開されている。またウェブリソースはマルチメディアであり、あらゆるコンテンツメディアを包括することができる。このようなウェブリソースをサイト間で合意の下に互いに容易に再利用できる環境があれば、ＷＷＷは格段に合理的で経済的なものになり、ＷＷＷの応用に大きな進歩をもたらすだろう。

【００１３】

例えば、本の売上情報やＴＶ番組の視聴率情報など、ウェブサイトを構成する情報資源の一部をアウトソーシングするといった、分散管理型のウェブサイト構築スタイルが可能となり、大きなウェブパーツ市場が生まれる可能性もある。また、各ショッピングサイトが抱える商品カタログを１つのウェブページ上で比較表示するショッピングモールや、複数の調達システムやオークションシステムなどが抱える案件を統合したマーケットプレースなどの仲介サービスを行うポータルサイトが最近次々と登場してきており非常に注目されている。これはウェブ情報が非常に氾濫してきている情勢においてウェブ情報を整理したり案内役を果たすサービスへ必然的なニーズが高まっているからであり、その要求に応える一つの形である。ウェブリソースを互いに再利用するための環境整備は、このようなポータルサイトの構築に大きな貢献をするだろう。その視点から、電子商取引システムなどウェブサイト間システム連携への足がかりとなる着実な技術移行という位置付けとも言える。

【００１４】

さて、ウェブページ検索サービスや各種商品比較サービスなど、複数のウェブサイトの情報を取りまとめる仲介サービスを行うポータルサイトが次々と登場し、非常に注目を集めているわけだが、このような仲介サービスは、さらに画像の収集やＭＰ３の収集など機能の専門化・多様化への発展を見せている。そのタスクの本質は、分散したウェブリソースを収集して加工した結果をウェブページとして提供するウェブサイト間のコンテンツ連携である。

【００１５】

ＨＴＭＬ技術では、ハイパーリンク機構を用いることにより任意のウェブページへジャンプできるようにしたり、フレーム機構を用いることにより複数のウェブページ全体を独立したウィンドウとして表示することはできるが、商品比較機能や合計値段見積もり機能の提供といった有機的なコンテンツの連携を行うにはまったく不十分である。これらを実現するためには、任意のウェブページを収集して柔軟に加工する機能が必要である。ＨＴＭＬのこのような機能欠如のため、ＣＧＩ（Ｃｏｍｍｏｎ　Ｇａｔｅｗａｙ　Ｉｎｔｅｒｆａｃｅ）やＳｅｒｖｌｅ

ｔなどのプログラム起動機構によって実行される外部プログラムやウェブサーバとは独立したデーモンプログラムにそれらの加工処理を行わせるという方法が取られている。この加工処理は概して次のような実行手続きが必要である。またデータベースを用いている場合は、さらにデータベースへのデータ登録や取出しの処理が加わる。

【００１６】

１．外部ウェブサイトのＨＴＭＬページを取得する処理

２．ＨＴＭＬページから必要なテキストを抽出する処理

３．抽出されたテキストを所望の形式に変換する処理

４．テキストをつなぎ合わせて１つのＨＴＭＬを作成する処理

このような解決手法には欠点がある。すなわち、これらの処理の多くは仲介サービス間で内容的に似通っているにもかかわらず、それぞれサイト構築者が１からプログラムを作成しているというのは生産効率および保守性が悪い。また、作成されたプログラムはそのサイトの環境に依存するものであり、必然的にそのサイト専用のプログラム資産となってしまうため、他のサイト環境において再利用することが出来ない。

【００１７】

このような欠点は、WWW技術においてコンテンツ連携をターゲットに置き、それを容易に実現するためのツールあるいはシステムが存在しないことが原因である。

【００１８】

【発明が解決しようとする課題】

このように、従来は、複数のウェブページから必要とする情報を収集して、それを特定の書式に変換するといった加工を行った後、１つのウェブページ上に合成するための汎用的な手法がないという問題点があった。

【００１９】

今後、複数のウェブサイトの情報をとりまとめるポータルサイトのような仲介サービスがより活発化する状況下において、コンテンツ連携に特化した共通のプラットフォームを提供することは、生産効率およびポータビリティの面で有効な

手段の１つである。

【００２０】

そこで、本発明は、上記問題点に鑑み、複数のウェブサイトの情報を１つのウェブ文書上に合成することが容易にしかも汎用的に行える文書合成方法およびそれを用いた文書合成装置を提供することを目的とする。

【００２１】

【課題を解決するための手段】

本発明は、インターネットにおけるWWW（Ｗｏｒｌｄ　Ｗｉｄｅ　ｗｅｂ）上のマークアップ言語で記述された複数の第１の文書の内容の一部をWWW上のマークアップ言語で記述された第２の文書に合成するためのものであって、前記第１の文書の該インターネット上の所在と、該第１の文書から抽出する部分文書の範囲と、前記第２の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第２の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第２の文書に従って、前記第１の文書から前記部分文書を抽出して、その部分文書を前記第２の文書上の前記指定された挿入位置に挿入するとともに、前記変換ルールを用いて前記第２の文書上の前記指定された範囲の文書構造を変換することを特徴とする。

【００２２】

本発明によれば、複数のウェブサイトの情報を１つのウェブ文書上に合成することが容易にしかも汎用的に行える。

【００２３】

好ましくは、前記第２の文書は、前記第２の文書上の前記部分文書の挿入位置とを指定するとともに、前記第１の文書の所在と、該第１の文書から抽出する部分文書の範囲とを記述するため第１のタグ（挿入命令タグｐｚ：ｔａｒｇｅｔｓ）と、前記変換ルールを用いて文書構造を変換すべき範囲を指定するとともに、前記変換ルールを記述したファイルの識別情報を記述するための第２のタグ（変換命令タグｐｚ：ｃｏｎｖｅｒｔ）とを用いて記述されている。

【００２４】

また、好ましくは、前記第２の文書は、ＸＭＬ（Ｅｘｔｅｎｓｉｂｌｅ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）で記述されている。

【００２５】

さらに、好ましくは、前記第１の文書がＸＭＬで記述されていないときは、まず、ＸＭＬによる記述型式に変換した後、前記第１の文書から前記部分文書を抽出して、その部分文書を前記第２の文書上の前記指定された挿入位置に挿入する。

【００２６】

なお、上記手法をインターネット上のウェブサーバに組み込み、クライアント装置（ウェブブラウザ）から前記第２の文書の要求を受けたとき、この第２の文書にの記述に従って１または複数の部分文書を合成した第２の文書を要求元のウェブブラウザに提供するサーバ装置を構成することができる。

【００２７】

【発明の実施の形態】

以下、本発明の実施形態について図面を参照して説明する。

【００２８】

なお、以下の説明は、次に示す項目の順になされている。

【００２９】

（Ａ）複数のウェブサイトの情報を１つのウェブ文書に合成するために必要とされる機能

（Ｂ）ＸＭＬ－Ｐ’ｚ文書

（Ｂ－１）ＸＭＬ－Ｐ’ｚ言語の仕様

（Ｂ－２）ＸＭＬ－Ｐ’ｚ言語処理系の構成および動作

（Ｃ）複数のウェブ文書を１つのウェブ文書上に合成するための一連の動作

（Ｄ）ウェブ文書の合成処理のためのＸＭＬ－Ｐ’ｚサーバ間の協調動作

（Ｅ）追記

（Ａ）複数のウェブサイトの情報を１つのウェブ文書に合成するために必要とされる機能

まず、実施形態の説明する前に、複数のウェブサイトの情報（ウェブ文書）を１つのウェブ文書に合成するために必要とされる機能について説明する。

【００３０】

複数のウェブ文書を１つのウェブ文書上に合成するために必要な機能は、抽出・挿入・変換の３種類に絞り込まれる。ただし、ウェブサイトの情報、すなわち、コンテンツとしてのウェブ文書（例えばＨＴＭＬ文書）の全てが必要となるわけではなく、そのうちの一部のみが必要となるのが一般であることから、抽出機能には任意のウェブ文書のうちの部分文書を取り込むことが要求される。また、抽出された複数の部分文書を組み合わせて合成する際に、たとえば表の中に表を入れるというような柔軟な挿入機能が要求される。さらにそれだけでは不十分で、抽出してきた部分文書を一覧表型式に合成する際に、形式が不均一である場合に、それらを同じ形式に合わせるというように、文書の変換機能が要求されることもある。

【００３１】

この分析に基づき、本発明は、次のような記述モデルを採用する。まず、ＳＳＩ（Ｓｅｒｖｅｒ　Ｓｉｄｅ　Ｉｎｃｌｕｓｉｏｎ）およびその発展系であるＡＳＰ（Ａｃｔｉｖｅ　Ｓｅｒｖｅｒ　Ｐａｇｅｓ）やＪＳＰ（Ｊａｖａ　Ｓｅｒｖｅｒ　Ｐａｇｅｓ）と同じように、複数のウェブ文書（部分文書）を合成するための合成用ウェブ文書内の任意位置にコマンドを配置し、そのコマンド実行結果が当該位置に埋め込まれるという、パッチワーク的な文書処理方式を採用する。

【００３２】

そして、用意するコマンドとして、どのウェブページのどの部分を抽出してどこに挿入するのかを示す部分文書の挿入コマンドを用意する。この方法は、抽出される部分文書の指定とその挿入位置を骨格となる合成用ウェブ文書を用いて自由にそして感覚的に記述できる利点がある。それに加えて、骨格となる合成用ウェブ文書の任意の範囲に対して、変換処理を施すことができる変換コマンドを用意する。この変換コマンドは、範囲情報と変換ルールを入力とし変換結果の文書を出力とする。まとめると、合成用ウェブ文書内の任意の位置に合成ロジックを

埋め込むことが出来る記述形式を採用し、合成ロジック用コマンドとして挿入および変換を用意した。

【００３３】

また、採用した実行モデルの１つはＳＳＩと同様であり、この合成用ウェブ文書をウェブサーバに配置しておき、ブラウザからそのＵＲＬへの要求があった場合に、そのウェブサーバに配置された言語処理系がその合成用ウェブ文書に含まれるコマンドを解釈実行し、その結果をブラウザに返すというものである。この方法では、サイト構築者は、合成用ウェブ文書をウェブサーバに配置しておくだけで解釈実行の起動について意識しなくてよいという利点がある。ただし、そのような実行方法だけではなく、ユーザが手動で解釈実行を行わせることも原理的に可能である。この場合、クライアント側で任意の合成を行うことができる。

【００３４】

さて、このような合成用ウェブ文書の記述においてＸＭＬ（Ｅｘｔｅｎｓｉｂｌｅ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ）は最適な言語である。ＸＭＬはタグ名や属性名を自由に定義し、それに対してアプリケーション側がセマンティクスを与えることが出来る。それに加えて、またＸＭＬはツリー型の文書構造を持つことが保証されているため、ツリー構造で表現される文書構造上における１つのノードとして表される特定のエレメントを指し示すだけで部分文書（文書範囲）を指定することができる。

【００３５】

また、ＸＭＬ自体はローレベルでの標準のデータ形式としての需要から、ＸＳＬＴ（Ｅｘｔｅｎｓｉｂｌｅ　Ｓｔｙｌｅｓｈｅｅｔ　Ｌａｎｇｕａｇｅ　Ｔｒａｎｓｆｏｒｍａｔｉｏｎｓ）（参考文献：ｈｔｔｐ：／／ｗｗｗ．ｗ３．ｏｒｇ／ＴＲ／ｘｓｌｔ）などの変換系技術も整備されているし、今後のＸＭＬ技術の発展においても上記の合成用ウェブ文書を、このＸＭＬ言語を応用した言語（本発明に係るＸＭＬ応用言語）で記述することで拡張性およびツール利用などの利便性が約束されることになる。

【００３６】

また、将来、ＨＴＭＬ文書だけでなくＸＭＬ文書がよく用いられるようになっ

たときにも、抽出対象として扱いやすいという利点がある。

【００３７】

そこで、本発明では、合成用ウェブ文書の記述言語をＸＭＬ応用言語として具体的に設計する。

【００３８】

本発明では、結合のためのベースとなる合成用ウェブ文書（合成用ウェブページと呼ぶこともある）をＸＭＬで記述し、指定した他のウェブ文書から指定した範囲の部分（部分文書）を抽出して、それを合成用ウェブ文書の指定された位置に挿入し、合成用ウェブ文書の指定した範囲に変換処理（所望の文書構造への変換処理）を施す、挿入・変換の２つの合成ロジック命令をその合成用ウェブ文書内にエレメントとして持たせる方針を採る。

【００３９】

このような合成用ウェブ文書、すなわち、ＸＭＬ文書（ＸＭＬページ）を、ここでは、ＸＭＬ－Ｐ’ｚ（ＸＭＬ－Ｐｉｅｃｅｓ）文書（ＸＭＬ－Ｐ’ｚページ）と呼ぶものとする。

【００４０】

ＸＭＬ－Ｐ’ｚ言語処理系をウェブサーバへ組み込みむことにより、図１に示すような動作が可能になる。なお、ＸＭＬ－Ｐ’ｚ言語処理系を組み込んだウェブサーバをＸＭＬ－Ｐ’ｚサーバと呼ぶこともある。具体的には、Ｍｉｃｒｏｓｏｆｔ社のウェブサーバであるＩＩＳ（Ｉｎｔｅｒｎｅｔ　Ｉｎｆｏｒｍａｔｉｏｎ　Ｓｅｒｖｅｒ）への組み込む場合を例にとり説明する。

【００４１】

図１に示した基本的な動作原理において、

（ステップＳ１０１）クライアント端末Ｂ１のウェブブラウザからＸＭＬ－Ｐ’ｚサーバＡ１（以下、簡単にサーバＡ１と呼ぶ）へのＸＭＬ－Ｐ’ｚ文書２の要求（ＧＥＴ／ＨＴＴＰ）が送信される。

【００４２】

（ステップＳ１０２）サーバＡ１は、要求されたリソースがＸＭＬ－Ｐ’ｚ文書かどうかを判断する。

【００４３】

（ステップＳ１０３）ＸＭＬ－Ｐ’ｚ文書と判断した場合、サーバＡ１は、ＸＭＬ－Ｐ’ｚ言語処理系（図１の合成処理部１）を起動し、ＸＭＬ－Ｐ’ｚ文書２に記述されている、指定されたウェブサーバ（例えば、ここでは、ウェブサーバＡ２、Ａ３）のウェブ文書（ページ）Ｗ２、Ｗ３から指定した範囲の部分（部分文書）を抽出し、それをＸＭＬ－Ｐ’ｚ文書の指定位置に挿入するとともに、ＸＭＬ－Ｐ’ｚ文書に記述されている指定された範囲に変換処理を施す。最終的に、ＸＭＬ－Ｐ’ｚ言語処理系の処理結果としてのＸＭＬ文書（合成されたウェブ文書）Ｗ１を得る。

【００４４】

（ステップＳ１０４）得られたＸＭＬ文書を要求元への返答としてブラウザに送信する。

【００４５】

上記動作は、ウェブサーバの設定によって実現する。ほとんどのウェブサーバには、ＵＲＬ文字列のパターン（よくあるのがオブジェクトの拡張子）とそれを前処理するのに必要なアドインを対応付ける機能を持っており、それを利用することにより（ステップＳ１０２）～（ステップＳ１０３）を実現できる。

【００４６】

また、ウェブブラウザがＸＭＬ文書を表示できる場合はＸＭＬ文書を、表示できない場合はサーバＡ１側でスタイルシートを処理してＨＴＭＬ文書を返すという処理があってもよい。

【００４７】

（Ｂ）ＸＭＬ－Ｐ’ｚ文書

ＸＭＬ－Ｐ’ｚ文書では、挿入命令エレメント「ｐｚ：ｔａｒｇｅｔｓ」と変換命令エレメント「ｐｚ：ｃｏｎｖｅｒｔ」とを定義する。

【００４８】

挿入命令タグを用いることにより、ＸＭＬ－Ｐ’ｚ文書のツリー構造で表現される文書構造上における１つのエレメント下の子文書として他のＸＭＬ文書また

はＨＴＭＬ文書の部分文書を挿入（合成）することができる。挿入対象とする部分文書の指定としては、ＸＰｏｉｎｔｅｒ付ＵＲＬ（参考文献：ｈｔｔｐ：／／ｗｗｗ．ｗ３．ｏｒｇ／ＴＲ／ＷＤ－ｘｐｔｒ＃ｕｒｉ－ｅｓｃａｐｉｎｇ）を採用する。これにより１行で簡潔に特定ウェブページの部分文書を指定することが出来る。ただしＸＰｏｉｎｔｅｒ規格はＸＭＬのためのものであるため、ＨＴＭＬを直接対象とすることが出来ない。このことから、抽出する際に、ＨＴＭＬ－ＤＯＭ（Ｄｏｃｕｍｅｎｔ　Ｏｂｊｅｃｔ　Ｍｏｄｅｌ）およびＸＭＬ－ＤＯＭを用いることにより、構造的に等価なＨＴＭＬ－ＸＭＬ変換を行う機構を導入する。これによりＨＴＭＬ文書はＸＭＬ文書として扱うことが出来るので、すべての加工処理はＸＭＬとして行うことが出来るようになる。

【００４９】

またＸＭＬ－Ｐ’ｚ文書では、変換命令エレメントを用いることにより、任意のエレメント（ノード）下の各子文書に対してＸＳＬＴ（Ｅｘｔｅｎｓｉｂｌｅ　Ｓｔｙｌｅ　Ｌａｎｇｕａｇｅ　ｔｒａｎｓｆｏｒｍａｔｉｏｎｓ）を用いた変換操作を実行することができる。すなわち、変換命令エレメントによって指示された、変換命令エレメントの子ノードとして配置される各子文書に対して指定されたＸＳＬＴが適用される。これを利用して、挿入命令タグによって挿入されたウェブ文書を変換命令タグを用いて変換することができる。

【００５０】

以下は、挿入命令エレメントと変換命令エレメントとを用いた、挿入機能と変換機能を有するＸＭＬ－Ｐ’ｚ文書の単純な例である。

【００５１】

（ＸＭＬ－Ｐ’ｚ文書の第１の例）

```
1. <?xml version=" 1.0" ?>
2. <root xmlns:pz=" http://www.shiba.co.jp/xmlpz" >
3.  <category>xxx</category>
4.  <item_holder>
5.   <pz:convert href=" xxx.xsl" >
6.    <pz:targets href=" http://www.yyy.com/index.xml#xpointer(//item)
```

”/>

７．　</pz:convert>

８．</item_holder>

９．</root>

図１１（ａ）は、上記第１の例の文書構造を模式的に示したもので、図１１（ｂ）は、上記第１の例を解釈した後のＸＭＬ文書の文書構造を模式的に示したものである。

【００５２】

上記第１の例において、６行目の挿入命令エレメント「ｐｚ：ｔａｒｇｅｔｓ」で指定された挿入対象の各ＸＭＬ部分文書（ｈｔｔｐ：／／ｗｗｗ．ｙｙｙ．ｃｏｍ／ｉｎｄｅｘ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（／／ｉｔｅｍ）で、以下、簡単に部分文書ＰＤ１と呼ぶ）が、５行目の変換命令エレメント「ｐｚ：ｃｏｎｖｅｒｔ」で指定されたＸＳＬＴの変換ルールが適用されて変換され、４行目～８行目にある「ｉｔｅｍ＿ｈｏｌｄｅｒ」エレメントの子エレメントとして、図１１（ｂ）に示すように、挿入される。ただし、６行目の「ｐｚ：ｔａｒｇｅｔｓ」で指定されているウェブ文書はＸＰｏｉｎｔｅｒにマッチするすべての部分文書であり（上記第１の例の場合は、「ｉｔｅｍ」タグがルートとなる部分文書すべて）、一般的には複数のウェブ文書となる。

【００５３】

上記の分散ウェブリソースのウェブ文書合成手法は以下の優位性がある。

【００５４】

優位点の一つは構築容易性である。本手法は、データベースを中心とした従来の方式と異なり、情報資源の合成ロジックをプログラミング言語なしで簡潔に記述できるので、ウェブ文書統合の構築・構成変更が容易である。またブラウザからの要求時に解釈処理されるインタプリタ型の実行モデルが採用されているので、合成ロジックの変更はただちに反映される。

【００５５】

もう一つの優位点は高い再利用性にある。ＸＭＬ－Ｐ’ｚのフレームワークでは、コンテンツ・変換ルール・合成ロジックなどすべての構成要素がウェブリソ

ースとして提供される。ウェブ文書の外にプログラムとして合成ロジックを持たせていた従来の方法と異なり、本方式ではＵＲＬを介してこれらすべての構成要素にアクセスすることができるので、原理的に世界中のウェブシステムから再利用することができる。このことはウェブサイトを越えた分散システムに必要な各リソースを自由に配置することを意味し、運用に応じた柔軟なシステム構築および変更が可能となる。

【００５６】

さらにＸＭＬ－Ｐ’ｚ文書が別サイトのＸＭＬ－Ｐ’ｚ文書を合成対象とすることでウェブサイト間で合成ロジックを分業（連携）することができる。

【００５７】

またＨＴＴＰ以外の特別なプロトコルをまったく用いておらず、ウェブリソースを提供する側ウェブサイトは特別な処理システムを導入する必要がない。したがってあらゆるウェブサイトの情報資源を再利用対象とすることができる。言い換えれば、既存のウェブサイトはシステム資源をそのまま生かすことが出来、ＸＭＬ－Ｐ’ｚ資源を別途作成するだけで合成することが出来る。

【００５８】

ただし、このような高いアクセシビリティについては、著作権問題など利用に関する実運用上の問題がからむ。たとえば、ＸＭＬ－Ｐ’ｚ技術を用いれば、ウェブ検索サービスを行っている複数のウェブサイトの検索結果を合成するメタ検索ページを提供することが簡単にできるが、著作権問題に抵触する。このような問題は、現在のＷＷＷにおいてもハイパーリンクの許可をめぐって問題となっており運用で乗り切っている現状がある。これに対して、　Ｅｘｔｒａｎｅｔ構築技術などアクセスコントロールに関するＷＷＷ技術が提供されている一方、ＷＷＷで公開された著作物の取り扱いに関する法整備が急ピッチで行われているところである。またＸＭＬ－Ｐ’ｚフレームワークにおいても、将来の課題として著作権問題を包括的に取り扱うモデルを導入したいと考えている。

【００５９】

次に、以上、説明した分散ウェブリソースのウェブ文書合成手法を次の２つのパートに分けて説明する。

【００６０】

（Ｂ－１）　ＸＭＬ－Ｐ’ｚ言語の仕様

（Ｂ－２）　ＸＭＬ－Ｐ’ｚ言語処理系の構成および動作

ＸＭＬ－Ｐ’ｚ言語とは、合成ロジックを含むウェブページ記述言語であり本システムの中核をなす。まずその言語仕様について（Ｂ－１）で説明する。次にＸＭＬ－Ｐ’ｚ言語で記述されたＸＭＬ－Ｐ’ｚ文書を解釈処理し、その結果を返す言語エンジンとしての言語処理系の構成およびその動作について（Ｂ－２）で説明する。

【００６１】

（Ｂ－１）ＸＭＬ－Ｐ’ｚ言語の仕様

ＸＭＬ－Ｐ’ｚ言語とは、特定のタグ名に対してセマンティクスが与えられたＸＭＬ応用言語の１つであり、分散ウェブリソースの合成を目的としたウェブ文書記述言語である。通常のＸＭＬ文書と同様、コンテンツを記述することができるのに加え、任意のエレメントに対して、ウェブリソースを操作する命令用のタグ名を記述することにより、合成ロジックを内部に含めることができる。この合成ロジックの記述はＨＴＭＬのハイパーリンクのように簡潔である。

【００６２】

このように合成ロジックを含むＸＭＬ－Ｐ’ｚ言語にて記述されたＸＭＬ－Ｐ’ｚ文書は、その合成ロジックに従い仮想的に分散リソースを統合・合成したウェブ文書へと解釈される。

【００６３】

ウェブリソース操作に関する命令エレメントとして「ｔａｒｇｅｔｓ」および「ｃｏｎｖｅｒｔ」の２つが用意されており、ＸＭＬネームスペースとして「ｐｚ」を予約している。これらの命令エレメントを組み合わせ用いることにより、他のウェブ文書を含めた任意の部分文書の抽出および自文書の挿入やＸＳＬＴを用いた構造変換を行うことができる。以下に各命令エレメント（ｐｚ：ｃｏｎｖｅｒｔエレメント、ｐｚ：ｔａｒｇｅｔｓエレメント）について説明する。

【００６４】

また、これらの命令エレメントは深さ優先の探索順序で解釈されなければなら

ない。たとえば、図１２に示すＸＭＬ－Ｐ’ｚ文書の文書構造において、ｐｚ：ｃｏｎｖｅｒｔエレメントの子エレメントとして、ｐｚ：ｔａｒｇｅｔｓエレメントが複数ある場合、各ｐｚ：ｔａｒｇｅｔｓエレメントが兄から弟へ順に解釈された後、ｐｚ：ｃｏｎｖｅｒｔエレメントが解釈される。

【００６５】

また、各命令タグの項でも説明しているとおり、挿入命令エレメントによって挿入されるウェブ文書および変換命令エレメントによって変換するウェブ文書は、合成、変換する前にＸＭＬ－Ｐ’ｚ文書として解釈されなければならない。すなわち、命令エレメントによって挿入、変換するウェブ文書内に命令エレメント（挿入、変換命令エレメント）が含まれている場合、それらが優先的に上述の順序で解釈されたのち、挿入先である本ＸＭＬ－Ｐ’ｚ文書の解釈実行が続行されるという再帰的な解釈処理の流れとなる。

【００６６】

また、ウェブリソースの指定子としてＸＰｏｉｎｔｅｒ付ＵＲＬを導入している。これはＸＰｏｉｎｔｅｒ規格（参考文献：ｈｔｔｐ：／／ｗｗｗ．ｗ３．ｏｒｇ／ＴＲ／ＷＤ－ｘｐｔｒ）に準拠するものであるが、本規格ではＸＰｏｉｎｔｅｒ付ＵＲＬの相対指定について未定義であるので、ＸＭＬ－Ｐ’ｚ言語では独自に規格を定めている。

【００６７】

以下にその規格を示す。

【００６８】

（ＸＭＬネームスペース）

ＸＭＬ－Ｐ’ｚの各命令タグを利用するためには、以下のネームスペースを宣言しなければならない。

【００６９】

・ネームスペース名

ｐｚ

・ネームスペースＵＲＩ

ｈｔｔｐ：／／ｓｈｉｂａ．ｃｏ．ｊｐ／ｘｍｌｐｚ

（ｐｚ：ｔａｒｇｅｔｓエレメント）

任意のウェブリソースを抽出・挿入する

文法

＜ｐｚ：ｔａｒｇｅｔｓ
ｈｒｅｆ＝”ｗｅｂ－ｒｅｓｏｕｒｃｅｓ－ｕｒｌ”＞
＜／ｐｚ：ｔａｒｇｅｔｓ＞

・属性

ｈｒｅｆ

挿入対象となる複数のウェブリソースへのＵＲＬ。ＵＲＬがＸＰｏｉｎｔｅｒ付である場合、ＵＲＬのボディ部のウェブ文書においてＸＰｏｉｎｔｅｒパターンにマッチするすべての部分文書が指定される。

【００７０】

・構造制約

親エレメント：　任意

子エレメント：　なし

・注釈

ｐｚ：ｔａｒｇｅｔｓエレメントは、ｈｒｅｆ属性によって指定された単数あるいは複数のウェブリソースをＸＭＬ－Ｐ’ｚ文書として解釈したのち当該エレメントのコンテクストに対して挿入し、ｐｚ：ｔａｒｇｅｔｓエレメント自身は消滅する。ｈｒｅｆ属性によって示されるＵＲＬがＸＰｏｉｎｔｅｒ付である場合、ＵＲＬのボディ部のウェブ文書においてＸＰｏｉｎｔｅｒパターンにマッチするすべての部分文書が指定される。

【００７１】

・サンプル

以下の例は、自文書内に含まれている本のデータに加え、「ｈｔｔｐ：／／ｗｗｗ．ｘｘｘ．ｃｏｍ／ｂｏｏｋｌｉｓｔ．ｘｍｌ」ページ内に含まれる本データをすべて取り込むＸＭＬ－Ｐ’ｚ文書である。

【００７２】

```
1. <?xml version=" 1.0" ?>
2. <bookstore specialty=" novel"
3.            xmlns:pz=" http://www.shiba.co.jp/xmlpz" >
4.  <book style=" textbook" >
5.   <author>
6.    <first-name>Shinichiro</first-name>
7.    <last-name>Hamada</last-name>
8.    <publication>Selected Short Stories of
9.     <first-name>Shinichiro</first-name>
10.     <last-name>Hamada</last-name>
11.    </publication>
12.   </author>
13.   <price>55</price>
14,  </book>
15.  <pz:targets href=" http://www.xxx.com/booklist.xml#xpointer(//boo
k)" />
16. </bookstore>
```

（ｐｚ：ｃｏｎｖｅｒｔエレメント）

任意の部分文書群をＸＳＬＴ文書を用いて変換する

文法

＜ｐｚ：ｃｏｎｖｅｒｔ
ｈｒｅｆ＝”ｘｓｌｔ－ｕｒｌ”＞
＜／ｐｚ：ｔａｒｇｅｔｓ＞

属性

ｈｒｅｆ

変換ルールを定義するＸＳＬＴ文書へのＵＲＬ。ＵＲＬがＸＰｏｉｎｔｅｒ付である場合、ＵＲＬのボディ部のウェブ文書においてＸＰｏｉｎｔｅｒパタ

ーンにマッチする部分文書のうち、文書順で先頭の部分文書が指定される。

【００７３】

構造制約

親エレメント：　任意

子エレメント：　任意

注釈

ｐｚ：ｃｏｎｖｅｒｔエレメントは、当該エレメント下の各子文書それぞれに対して、ｈｒｅｆ属性によって指定されたＸＳＬＴ文書を適用して変換する。変換された各子文書は、ＸＭＬ－Ｐ’ｚ文書として解釈した後ｐｚ：ｃｏｎｖｅｒｔエレメントのコンテクストに挿入され、ｐｚ：ｃｏｎｖｅｒｔエレメント自身は消滅する。ｈｒｅｆ属性によって示されるＵＲＬがＸＰｏｉｎｔｅｒ付である場合、ＵＲＬのボディ部のウェブ文書においてＸＰｏｉｎｔｅｒパターンにマッチする部分文書のうち、文書順で先頭の部分文書が指定される。

【００７４】

サンプル

以下の例は、「ｔｅｘｔｂｏｏｋ」エレメントで表現されている自文書内に含まれている教科書データに加え、「ｈｔｔｐ：／／ｗｗｗ．ｘｘｘ．ｃｏｍ／ｂｏｏｋｌｉｓｔ．ｘｍｌ」ページ内に含まれるすべての教科書データを「ｔｅｘｔｂｏｏｋ－ｂｏｏｋ．ｘｓｌ」というＸＳＬＴ文書に記述された変換ルールに従って、共通書籍形式へ変換し、また、「ｈｔｔｐ：／／ｗｗｗ．ｙｙｙ．ｃｏｍ／ｉｎｄｅｘ．ｈｔｍｌ」ページで公開されている本データを共通書籍形式へ変換したものをすべて取り込むＸＭＬ－Ｐ’ｚ文書である。

【００７５】

```
1. <?xml version=" 1.0" ?>
2. <bookstore specialty=" novel"
        xmlns:pz=" http://www.shiba.co.jp/xmlpz" >
3.   <pz:convert href=" textbook-book.xsl" >
4.     <textbook>
5.       <author>
```

```
６．        <first-name>Shinichiro</first-name>
７．        <last-name>Hamada</last-name>
８．        <publication>Selected Short Stories of
９．          <first-name>Shinichiro</first-name>
１０．          <last-name>Hamada</last-name>
１１．        </publication>
１２．      </author>
１３．      <price>55</price>
１４．    </textbook>
１５．    <pz:targets href=" http://www.xxx.com/booklist.xml#xpointer(//
textbook)" />
１６．  </pz:convert>
１７．  <pz:convert href=" html-book.xsl" >
１８．    <pz:targets href=" http://www.yyy.com/index.html#xpointer(//TA
BLE[2]//TR)" />
１９．  </pz:convert>
２０．</bookstore>
```

（ＸＰｏｉｎｔｅｒ付ＵＲＬの相対指定）

ウェブリソースが他のウェブリソースを参照指定する際に、自ウェブリソースの持つＵＲＬをベースとして相対的なＵＲＬを用いることができる。これを相対ＵＲＬと言う。資源を一意に区別するためには、処理系が相対ＵＲＬを絶対ＵＲＬへ展開しなければならない。その解決方法を以下に示す。ただし以下の説明において、用語はＩＥＴＦ（ｈｔｔｐ：／／ｗｗｗ．ｉｅｔｆ．ｏｒｇ／ｒｆｃ／ｒｆｃ１７３８．ｔｘｔ）に基づくものとする。

【００７６】

１．）ベースＵＲＬのオブジェクトと相対ＵＲＬのオブジェクトが異なる場合

ベースＵＲＬから（もしあれば）ＸＰｏｉｎｔｅｒフラグメントを取り除いたボディ部と、相対ＵＲＬから（もしあれば）ＸＰｏｉｎｔｅｒフラグメントを

取り除いたボディ部との間で、ＩＥＴＦ（ｈｔｔｐ：／／ｗｗｗ．ｉｅｔｆ．ｏｒｇ／ｒｆｃ／ｒｆｃ１８０８．ｔｘｔ）に基づいた相対ＵＲＬの解決を行った結果に対して、（もしあれば）相対ＵＲＬのＸＰｏｉｎｔｅｒフラグメントを与える。なお、ＸＰｏｉｎｔｅｒフラグメントとは、例えば、以下のサンプルの記述における「＃ｘｐｏｉｎｔｅｒ」以下の部分で、「＃ｘｐｏｉｎｔｅｒ（／ｎｏｄｅ１／ｎｏｄｅ２）」や、「＃ｘｐｏｉｎｔｅｒ（．／ｎｏｄｅ３／／ｎｏｄｅ４）」である。

【００７７】

・サンプル

（ベースＵＲＬ）　ｈｔｔｐ：／／ａａａ．ｃｏｍ／ｄｉｒ１／ｘｘｘ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（／ｎｏｄｅ１／ｎｏｄｅ２）

（相対ＵＲＬ）　　．／ｄｉｒ２／ｙｙｙ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（．／ｎｏｄｅ３／／ｎｏｄｅ４）

（解決結果）　ｈｔｔｐ：／／ａａａ．ｃｏｍ／ｄｉｒ１／ｄｉｒ２／ｙｙｙ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（．／ｎｏｄｅ３／／ｎｏｄｅ４）

２．）ベースＵＲＬのオブジェクトと相対ＵＲＬのオブジェクトが同じ場合

ベースＵＲＬがＸＰｏｉｎｔｅｒフラグメントを含んでいる場合はＸＰｏｉｎｔｅｒが示す文書ノード、ＸＰｏｉｎｔｅｒフラグメントを含んでいない場合はルート文書ノードを起点として、（もしあれば）相対ＵＲＬのＸＰｏｉｎｔｅｒの示すノードを決定し、そのノードパスを示すＸＰｏｉｎｔｅｒフラグメントを当該オブジェクトのＵＲＬに与える。

【００７８】

・サンプル

（ベースＵＲＬ）　ｈｔｔｐ：／／ａａａ．ｃｏｍ／ｄｉｒ１／ｘｘｘ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（／ｎｏｄｅ１／ｎｏｄｅ２）

（相対ＵＲＬ）　　ｈｔｔｐ：／／ａａａ．ｃｏｍ／ｄｉｒ１／ｘｘｘ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（．／ｎｏｄｅ３／／ｎｏｄｅ４）

（解決結果）　ｈｔｔｐ：／／ａａａ．ｃｏｍ／ｄｉｒ１／ｘｘｘ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（／ｎｏｄｅ１／ｎｏｄｅ２／ｎｏｄｅ３／／ｎｏｄｅ４）

３．）相対ＵＲＬにおいてオブジェクトが無指定である場合

ベースＵＲＬがＸＰｏｉｎｔｅｒフラグメントを含んでいる場合はＸＰｏｉｎｔｅｒが示す文書ノード、ＸＰｏｉｎｔｅｒフラグメントを含んでいない場合はルート文書ノードを起点として、（もしあれば）相対ＵＲＬのＸＰｏｉｎｔｅｒの示すノードを決定し、そのノードパスを示すＸＰｏｉｎｔｅｒフラグメントをベースＵＲＬのオブジェクトのＵＲＬに与える。

【００７９】

サンプル

（ベースＵＲＬ）　ｈｔｔｐ：／／ａａａ．ｃｏｍ／ｄｉｒ１／ｘｘｘ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（／ｎｏｄｅ１／ｎｏｄｅ２）

（相対ＵＲＬ）　　＃ｘｐｏｉｎｔｅｒ（．／ｎｏｄｅ３／／ｎｏｄｅ４）

（解決結果）　ｈｔｔｐ：／／ａａａ．ｃｏｍ／ｄｉｒ１／ｘｘｘ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（／ｎｏｄｅ１／ｎｏｄｅ２／ｎｏｄｅ３／／ｎｏｄｅ４）

（Ｂ－２）ＸＭＬ－Ｐ’ｚ言語処理系の構成および動作

次に、ＸＭＬ－Ｐ’ｚ言語の解釈処理系について説明する。

【００８０】

ＸＭＬ－Ｐ’ｚ言語処理系は、ＸＭＬ－Ｐ’ｚ文書の所在を示すＵＲＬまたはソースを入力とし、その解釈結果のＸＭＬ文書ソースを出力とするソフトウェアコンポーネントである。本処理系ではＸＭＬ－Ｐ’ｚ言語の解釈処理を２パスで行う方式を取っており、１パス目でＸＭＬとして構文解析を行ってＸＭＬ－ＤＯＭツリーを作成し、続いて２パス目でＸＭＬ－ＤＯＭツリーを深さ優先でたどりながら、ＸＭＬ－Ｐ’ｚ言語特有の命令エレメント（挿入、変換命令タグで囲まれた部分）の解釈処理を行う。この言語処理に際して、文法逸脱を発見した場合やネットワークトラブルなどのランタイムエラーが発生した場合でも、解釈処理をそのまま続行することにより、可能な最良の結果を出力する処理方針をとる。

【００８１】

またＸＭＬ－Ｐ’ｚ言語ではＸＰｏｉｎｔｅｒ付ＵＲＬを用いたウェブリソース指定が可能であるが、本処理系では、ＵＲＬで示される文書全体をダウンロー

ドした上で、ＸＰｏｉｎｔｅｒで指定された部分文書を切り出すという２段階の処理を行う方式を取る。これにより、ＸＰｏｉｎｔｅｒ付ＵＲＬに対応していないほとんどのウェブサーバに対しても、ウェブリソースを要求することが出来る。

【００８２】

以上が基本的な処理方針である。この処理方針に基づいた本処理系のシステム構成例について説明する。

【００８３】

図２は、ＸＭＬ－Ｐ’ｚ言語処理系１００（図１の合成処理部１に相当）の全体の構成例である。図２において、この言語処理系１００は、大きく分けて、ＸＭＬ－Ｐ’ｚ文書読込に関する処理モジュールである、解釈バッファファクトリ１０１と、読み込まれた文書を解釈した結果のＸＭＬを返す処理モジュールである、インタプリタ１０２の２つから構成されている。これらは基本的に独立に動作する。なお、図２中の２つの解釈バッファファクトリ１０１は同一物であるが見やすくするため分けて書いている。

【００８４】

解釈バッファファクトリ１０１は、ＸＭＬ－Ｐ’ｚ文書の所在を示すＵＲＬまたはソースの入力をトリガとして動作を開始し、まず、ＸＭＬノーマライザ１１１において、入力文書がＸＭＬならばそのまま、ＨＴＭＬならば同等の構造を持つＸＭＬへの等価変換処理を行った上で、ＸＭＬ－ＤＯＭパーサ１１４を用いてＸＭＬ－ＤＯＭツリーを作成し、さらに、ＸＰｏｉｎｔｅｒプロセッサ１１５において、ＵＲＬ内に含まれるＸＰｏｉｎｔｅｒフラグメントにしたがって部分文書を抽出した結果をもとに、解釈バッファイニシャライザ１１６は、解釈バッファ１０３，１０４を生成する。

【００８５】

さらに、ＵＲＬまたはソースの入力が処理系１００外部からであった場合、生成する解釈バッファを、デフォルト解釈バッファ１０３として登録する。ここで解釈バッファとはＸＭＬ－Ｐ’ｚ言語解釈処理の状態記憶でありインタプリタ１０２の解釈処理中に繁茂に更新される。

【００８６】

一方、インタプリタ１０２は処理系１００外部からの解釈結果の要求があった場合に動作を開始し、デフォルト解釈バッファ１０３の解釈用ＸＭＬ－ＤＯＭツリー１３１を深さ優先でたどりながら、ｐｚ：ｔａｒｇｅｔｓエレメントおよびｐｚ：ｃｏｎｖｅｒｔエレメントの２つの命令エレメントの解釈実行を行い、最終的に得られた解釈結果のＸＭＬ文書を出力する。

【００８７】

ただし、命令エレメントの解釈中に一時的に生成される部分文書をＸＭＬ－Ｐ’ｚ解釈処理するため、解釈バッファファクトリ１０１を用いて、一時解釈バッファ１０４を生成する。

【００８８】

次に、解釈バッファファクトリ１０１を構成する各構成部（モジュール）の処理動作を説明する。

【００８９】

解釈バッファファクトリ１０１を構成する、ＸＭＬノーマライザ１１１は、ＨＴＭＬ判定器１１２、および、ＨＴＭＬ－ＸＭＬコンバータ１１３から構成される。

【００９０】

ＨＴＭＬ判定器１１２は、与えられたＵＲＬが指し示すウェブリソース（ウェブ文書）がＨＴＭＬ文書かＸＭＬ文書かを判定する。その判定にはＨＴＴＰヘッダの「Ｃｏｎｔｅｎｔ－ｔｙｐｅ」を用いる方法とＵＲＬ内に含まれる拡張子を用いる方法の２段階のテストを行う。この処理動作を図３に示す。

【００９１】

図３において、まず、「Ｃｏｎｔｅｎｔ－Ｔｙｐｅ」を取得する（ステップＳ１）。この取得の方法として当該ＵＲＬに対して、ＨＥＡＤ要求を行うのがもっとも直接的である。しかしＨＥＡＤ要求を理解できないウェブサーバも世の中にたくさんある。代用としてＧＥＴ要求を用いることもできる。次に、当該ＵＲＬに対してＨＴＴＰ接続できたかどうか判定する（ステップＳ２）。もし接続に成功した場合は、ステップＳ３へ進み、失敗した場合はステップＳ５に進む。

【００９２】

ステップＳ３では、「Ｃｏｎｔｅｎｔ－Ｔｙｐｅ」ヘッダを取り出し、その中に「ｔｅｘｔ／ｈｔｍｌ」という文字列が含まれているか判定する。もし含まれていればＨＴＭＬと判定して終了し（ステップＳ６）、そうでなければ、ＸＭＬと仮判定して終了する（ステップＳ４）。

【００９３】

ステップＳ５では、ＵＲＬ内のオブジェクトフィールドの拡張子が「ｈｔｍｌ」または「ｈｔｍ」であるかどうか判定する。もしそうであればＨＴＭＬと判定して終了し（ステップＳ６）、そうでなければＸＭＬと仮判定して終了する（ステップＳ７）。

【００９４】

ＨＴＭＬ－ＸＭＬコンバータ１１３は、ＨＴＭＬ判定器１１２によってＨＴＭＬ文書と判断されたウェブリソースを構造的に等価なＸＭＬ文書へ変換する。これはＨＴＭＬ－ＤＯＭツリーからＸＭＬ－ＤＯＭツリーへと各ＤＯＭのメソッドを用いて順次移していくことで実現できる。ＨＴＭＬ－ＸＭＬコンバータ１１３の処理動作を図４に示す。

【００９５】

まず、ステップＳ１１において、与えられたＨＴＭＬ文書をＨＴＭＬパーサへ読み込ませ、ＨＴＭＬ－ＤＯＭツリーを構築する。ＨＴＭＬパーサはウェブブラウザが内部的に用いているものが望ましい。なぜならウェブブラウザが使用するＨＴＭＬパーサは、ＨＴＭＬ文法逸脱に対するエラーリカバリー機能がついているからである。

【００９６】

次に、ステップＳ１２において、ＸＭＬ－ＤＯＭパーサを用いて空のＸＭＬ－ＤＯＭツリーを構築する。そして、ステップＳ１３において、ＨＴＭＬ－ＤＯＭツリーを全探索しながら、立ち寄ったノードの値などを取り出しＸＭＬ－ＤＯＭツリーにノードとして挿入する。

【００９７】

以上の処理により、ＸＭＬノーマライザ１１１は、解釈バッファファクトリ１

０１にＵＲＬとして入力されたウェブリソースをすべてＸＭＬ文書として出力する。一方、ソースとして入力されたウェブリソースはすべてＸＭＬ文書と仮定して取り扱われる。

【００９８】

ＸＭＬノーマライザ１１１を通過したＸＭＬ文書またはソースとして入力されたＸＭＬ文書は、ＸＭＬ－ＤＯＭパーサ１１４に入力され、ＸＭＬ－ＤＯＭツリー化される。さらに、ＸＰｏｉｎｔｅｒプロセッサ１１５を用いて、ＵＲＬのＸＰｏｉｎｔｅｒフラグメントで示されているＸＭＬ文書内の部分文書のＸＭＬ－ＤＯＭツリーを得る。ＸＰｏｉｎｔｅｒプロセッサ１１５のＸＰｏｉｎｔｅｒフラグメントに対する処理動作を図５に示す。

【００９９】

まず、ステップＳ２１で、与えられたウェブリソースがＵＲＬによるものだったのか、ソースによるものだったのかを判定する。ソースによるものであった場合ＵＲＬは存在しないので、この時点で終了する。

【０１００】

次に、ステップＳ２２において、ＵＲＬのフラグメントからＸＰｏｉｎｔｅｒフラグメントを取り出す。ただしＸＰｏｉｎｔｅｒが指定されていなかった場合は空の文字列とする。続いて、ステップＳ２３においてＸＭＬ－ＤＯＭツリーのルートエレメントを基点としてＸＰｏｉｎｔｅｒが指し示すノードを同定する。これには一般的なＸＰｏｉｎｔｅｒ処理系を用いればよい。

【０１０１】

次に、ステップＳ２４において指し示されたノードがエレメントであるかどうかを判定する。もしエレメントでなければ異常終了する。続いて、ステップＳ２５において、得られたエレメントをルートエレメントとした部分文書のＸＭＬ－ＤＯＭツリーを切り出す。さらに、ステップＳ２６において、その切り出されたＸＭＬ－ＤＯＭツリーを新しいＸＭＬ文書のＸＭＬ－ＤＯＭツリーとする。

【０１０２】

さて、得られたＸＭＬ－ＤＯＭツリーを基に、解釈バッファイニシャライザ１１６は解釈バッファを生成する。このとき与えられたウェブリソースが言語処理

系１００外部からの入力によるものであった場合、その解釈バッファを、デフォルト解釈バッファ１０３として登録する。この解釈バッファ（メモリで構成されている）の初期化処理動作を図６に示す。なお、部分文書のＸＭＬ－ＤＯＭツリーの場合は、一時解釈バッファ１０４を図６と同様にして初期化する。

【０１０３】

まず、ステップＳ３１では、与えられたＸＭＬ－ＤＯＭツリーをソースＸＭＬ－ＤＯＭツリー１３４にコピーする。なお、ソースＸＭＬ－ＤＯＭツリー１３４は、以後のＸＭＬ－Ｐ’ｚ言語の解釈処理によって変更される前のＸＭＬ－ＤＯＭツリーの初期状態を記憶するバッファであり、ＸＭＬ－Ｐ’ｚ言語のソース提供などの用途を想定しているが、本実施形態では利用されない。

【０１０４】

次に、ステップＳ３２では、与えられたＸＭＬ－ＤＯＭツリーを解釈用ＸＭＬ－ＤＯＭツリー１３１へコピーする。解釈用ＸＭＬ－ＤＯＭツリー１３１は、インタプリタ１０２が解釈処理において構造の読み込みおよび解釈結果の書き込みに用いる。

【０１０５】

ステップＳ３３では、プログラムカウンタ１３２を解釈用ＸＭＬ－ＤＯＭツリー１３１のルートエレメントにセットする。プログラムカウンタ１３２は、インタプリタ１０２の解釈処理の進捗を記憶するポインタである。

【０１０６】

最後に、ステップＳ３４では、ロードフラグ１３３を「ｆａｌｓｅ」にセットする。ロードフラグ１３３とは、当該解釈バッファ１０３がすでに解釈処理済みかどうかを示すフラグである。インタプリタ１０２は、このフラグ１３３を利用して過去に解釈処理を施した解釈バッファについて解釈処理をし直さないようになっている。

【０１０７】

以上が、解釈バッファファクトリ１０１の処理動作の説明である。

【０１０８】

次に、インタプリタ１０２の処理動作について説明する。

【０１０９】

インタプリタ１０２を構成するコンテクストマネージャ１２１は、解釈処理において中心的役割を果たす。解釈バッファ１０３，１０４のプログラムカウンタ１３２，１４２に従い、解釈用ＸＭＬ－ＤＯＭツリー１３１，１４１の各ノードを深さ優先で立ち寄る際に、命令エレメントを発見すると該当する処理モジュール（ｔａｒｇｅｔｓコマンドプロセッサ１２２，ｃｏｎｖｅｒｔコマンドプロセッサ１２３）へ解釈処理を依頼する。命令エレメントの解釈処理が終了すると立ち寄り処理を続行する。すべての処理が終わると解釈結果としてＸＭＬ文書を出力する。この処理動作を図７に示す。以下、デフォルト解釈バッファ１０３を用いた解釈処理の場合を説明するが、一時解釈バッファ１０４の場合も同様である。

【０１１０】

まず、ステップＳ４１において、解釈バッファ１０３のロードフラグ１３３を調べる。ロードフラグが「ｔｒｕｅ」であればすでに解釈済みであり「ｆａｌｓｅ」ならば、まだ解釈処理が行われていない状態であることを意味する。「ｔｒｕｅ」ならば、ステップＳ４９へ進み、「ｆａｌｓｅ」ならば、ステップＳ４２へ進む。

【０１１１】

ステップＳ４２では、プログラムカウンタ１３２を読み込んで解釈処理対象とするエレメント（これをカレントエレメントと呼ぶ）を決定する。

【０１１２】

ステップＳ４３では、カレントエレメントのエレメント名が「ｐｚ：ｔａｒｇｅｔｓ」かどうかをチェックし、「ｐｚ：ｔａｒｇｅｔｓ」だった場合は、ステップＳ４へ進み、ｐｚ：ｔａｒｇｅｔｓエレメントの解釈処理をｔａｒｇｅｔｓコマンドプロセッサ１２２へ依頼する。

【０１１３】

続いて、ステップＳ４５では、カレントエレメントのエレメント名が「ｐｚ：ｃｏｎｖｅｒｔ」かどうかチェックし、「ｐｚ：ｃｏｎｖｅｒｔ」だった場合は、ステップＳ４６へ進み、ｐｚ：ｃｏｎｖｅｒｔエレメントの解釈処理をｃｏｎ

ｖｅｒｔコマンドプロセッサ１２３へ依頼する。

【０１１４】

続いて、ステップＳ４７で、深さ優先で移動先エレメントを決定しプログラムカウンタにセットする。カレントエレメントの子エレメントのうち、まだ解釈処理を行っていないエレメントがあれば、そのうちの長兄エレメントをプログラムカウンタへセットする。すべての子エレメントの解釈処理が行われているならば、親エレメントにプログラムカウンタへセットする。ただし親エレメントがいない場合は、プログラムカウンタを「ＮＵＬＬ」にセットする。

【０１１５】

ステップＳ８では、プログラムカウンタ１３２が「ＮＵＬＬ」かどうかをチェックし、「ＮＵＬＬ」でなければ、ステップＳ４２へ戻る。「ＮＵＬＬ」であれば、解釈用ＸＭＬ－ＤＯＭツリー１３１の解釈は終了したので、ステップＳ４９へ進む。

【０１１６】

ステップＳ４９では、ＸＭＬ－ＤＯＭパーサ１５１を用いて解釈バッファ１０３のＸＭＬ－ＤＯＭツリー１３１を基にＸＭＬ文書を生成し出力し、終了する。

【０１１７】

インタプリタ１０２を構成するｔａｒｇｅｔｓコマンドプロセッサ１２２は、ｐｚ：ｔａｒｇｅｔｓエレメントを解釈し、その結果をカレントエレメントに書き込む。この処理動作を図８に示す。

【０１１８】

まず、ステップＳ５１では、カレントエレメントであるｐｚ：ｔａｒｇｅｔｓエレメントのｈｒｅｆ属性値を取り出し、ステップＳ５２で、その属性値を解釈バッファファクトリ１０１の入力ＵＲＬとして、前述したＸＭＬノーマライザ１１１から解釈バッファイニシャライザ１１６による処理を経由して、一時解釈バッファ１０４を生成する。ただし、対象とするＵＲＬが相対ＵＲＬであった場合は、前述の「ＸＰｏｉｎｔｅｒ付ＵＲＬの相対指定」の説明に基づき、挿入先の解釈バッファのＵＲＬをベースとして絶対ＵＲＬへ変換する。

【０１１９】

次に、ステップＳ５３へ進み、生成された一時解釈バッファ１０４を、インタプリタ１０２を用いて解釈処理し、その結果としてのＸＭＬ文書を得る。

【０１２０】

最後に、ステップＳ５４では、ＤＯＭパーサ１５２を用いて、得られたＸＭＬ文書をＸＭＬ－ＤＯＭツリーに変換して、カレントエレメントである「ｐｚ：ｔａｒｇｅｔｓ」エレメントと入れ替える。また、生成した一時解釈バッファ１０４は破棄する。

【０１２１】

インタプリタ１０２を構成するｃｏｎｖｅｒｔコマンドプロセッサ１２３は、ｃｏｎｖｅｒｔエレメントを解釈し、その結果をカレントエレメントに書き込む。この処理動作を図９に示す。

【０１２２】

まず、ステップＳ６１では、カレントエレメントであるｐｚ：ｃｏｎｖｅｒｔエレメントのｈｒｅｆ属性値を取り出し、ステップＳ６２で、その属性値を解釈バッファファクトリ１０１の入力ＵＲＬとして、前述したＸＭＬノーマライザ１１１から解釈バッファイニシャライザ１１６による処理を経由して、一時解釈バッファ１０４を生成する。ただし、対象とするＵＲＬが相対ＵＲＬであった場合は、前述の（ＸＰｏｉｎｔｅｒ付ＵＲＬの相対指定）の説明に基づき、挿入先の解釈バッファのＵＲＬをベースとして絶対ＵＲＬへ変換する。

【０１２３】

次に、ステップＳ６３へ進み、生成された一時解釈バッファ１０４を、インタプリタ１０２を用いて解釈処理し、その結果としてＸＳＬＴ文書を得る。なお、このような処理を行うのは、ＸＳＬＴ文書自体がＸＭＬ－Ｐ’ｚ言語でかかれている可能性があるからである（すなわち合成結果としてＸＳＬＴ文書が構成されている可能性があるからである）。

【０１２４】

続いて、ステップＳ６４へ進み、ＸＳＬＴプロセッサ１２４により、カレントエレメントである「ｐｚ：ｃｏｎｖｅｒｔ」エレメントの子エレメントのうち、まだＸＬＳＴを適用していない長兄エレメント（およびその子孫エレメントを含

む部分文書）に、得られたＸＳＬＴ文書を用いて、当該部分文書の文書構造をＸＳＬＴ文書に記述された変換ルールを用いて変換し、その変換して得られたＸＭＬ－ＤＯＭツリーを、ステップＳ６５では、合成用ウェブ文書上の変換前の子エレメント（およびその子孫エレメントを含む部分文書）と入れ替える。

【０１２５】

ステップＳ６６において、もし未処理の子エレメントがあるならば、ステップＳ６４に戻る。すべての子エレメントが処理済ならば、ステップＳ６７へ進み、ｐｚ：ｃｏｎｖｅｒｔエレメントをｐｚ：ｃｏｎｖｅｒｔエレメントの各子部分文書である文書構造の変換されたものと入れ替える。

【０１２６】

以上が、インタプリタ１０２の処理動作であり、以上をもってＸＭＬ－Ｐ’ｚ言語処理系の各構成部についての説明は終了した。

【０１２７】

（Ｃ）複数のウェブ文書を１つのウェブ文書上に合成するための一連の動作

次に、図２に示した構成のＸＭＬ－Ｐ’ｚ言語処理系１００をウェブサーバへ組み込み、図１に示した基本的な動作を行って、実際に、ウェブサーバＡ２のウェブ文書Ｗ２からその一部を抽出し、その抽出された各部分文書を１つのウェブ文書上に合成し、合成されたウェブ文書（ＸＭＬ文書）Ｗ１を出力するための一連の動作を図１３～図１５に示すフローチャートを参照して説明する。

【０１２８】

ここで、合成用ウェブ文書としてのＸＭＬ－Ｐ‘ｚ文書２は、図１６に示すものであるとする。なお、図１６に示すＸＭＬ－Ｐ’ｚ文書は、図１のＸＭＬ－Ｐ‘ｚ文書２のうちの一部分を抜粋したものを示している。

【０１２９】

図１６に示すＸＭＬ－Ｐ‘ｚ文書は、「ｔｅｘｔｂｏｏｋ」エレメントＥ１で表現されている自文書内に含まれている教科書データと、ｐｚ：ｔａｒｇｅｔｓエレメントＥ２にて挿入される「ｈｔｔｐ：／／ｗｗｗ．ｘｘｘ．ｃｏｍ／ｂｏｏｋｌｉｓｔ．ｘｍｌ」のウェブ文書内に含まれるすべての教科書データとを、

「ｔｅｘｔｂｏｏｋ－ｂｏｏｋ．ｘｓｌ」というＸＳＬＴ文書に記述された変換ルールに従って、共通書籍形式へ変換して、合成されたウェブ文書（ＸＭＬ文書）Ｗ１を出力するためのものである。

【０１３０】

図１において、クライアント端末Ｂ１のウェブブラウザからＸＭＬ－Ｐ’ｚサーバＡ１（以下、簡単にサーバＡ１と呼ぶ）へのＸＭＬ－Ｐ’ｚ文書２の要求がなされたとする（ステップＳ２０１）。

【０１３１】

サーバＡ１の言語処理系１００は、要求された文書が自身が持つ合成用ウェブ文書（ＸＭＬ－Ｐ‘ｚ文書）２であるので、ＸＭＬ－ＤＯＭパーサ１１４を用いて当該ＸＭＬ－Ｐ‘ｚ文書のＸＭＬ－ＤＯＭツリーを作成する（ステップＳ２０２）。この作成されたＸＭＬ－ＤＯＭツリーの図１６に対応する部分は、例えば、図１７に示すものである。なお、図１７では、説明の簡単のために概略的に示している。

【０１３２】

この作成されたＸＭＬ－ＤＯＭツリーをデフォルト解釈バッファ１０３のソースおよび解釈用ＤＯＭツリー１３４，１３１にコピーし、その他、図６に示したようにして、デフォルト解釈バッファ１０３を初期化する（ステップＳ２０３）。

【０１３３】

次に、このデフォルト解釈バッファ１０３の解釈処理をインタプリタ１０２にて行う。ここで、例えば、図１７に示したようなＸＭＬ－ＤＯＭツリーを解釈するものとする。

【０１３４】

インタプリタ１０２は、前述したように、命令エレメントを深さ優先で移動先のエレメントを決定していくので、図１７に示すＤＯＭツリーにおいては、まず、ｐｚ：ｔａｒｇｅｔｓエレメントＥ２を解釈処理する（ステップＳ２０４～ステップＳ２０５）。その後、エレメントＥ１，Ｅ２の親エレメントであるｐｚ：ｃｏｎｖｅｒｔエレメントＥ３を解釈処理する（ステップＳ２０６～ステップＳ

２０７）。その後、図１７には示していないが、ｐｚ：ｃｏｎｖｅｒｔエレメントＥ３の弟エレメント、あるいは、親エレメントへ、プログラムカウンタ１３２を移動させて、プログラムカウンタが「ＮＵＬＬ」になるまで、このデフォルト解釈バッファ１０３の解釈処理を進めていく（ステップＳ２０８）。

【０１３５】

さて、ステップＳ２０５では、ｐｚ：ｔａｒｇｅｔｓエレメントＥ２の解釈処理を行うわけだが、ここでの処理動作を図１４に示す。

【０１３６】

ｔａｒｇｅｔｓコマンドプロセッサ１２２は、ｐｚ：ｔａｒｇｅｔｓエレメントＥ３のｈｒｅｆ属性値、すなわち、「ｈｔｔｐ：／／ｗｗｗ．ｘｘｘ．ｃｏｍ／ｂｏｏｋｌｉｓｔ．ｘｍｌ＃ｘｐｏｉｎｔｅｒ（／／ｔｅｘｔｂｏｏｋ）」を取り出し、その属性値を解釈バッファファクトリ１０１の入力ＵＲＬとする。ＸＭＬノーマライザ１１１は、この入力ＵＲＬにて指定された文書がＸＭＬ文書でないならそれをＸＭＬ文書に変換した後（ステップＳ２１２）、ＸＭＬ－ＤＯＭパーサ１１４にて、このＸＭＬ文書のＸＭＬ－ＤＯＭツリーを作成する（ステップＳ２１３）。なお、ここでは、当該指定された文書はＸＭＬ文書であるので、そのまま、ＸＭＬ－ＤＯＭパーサ１１４にて、このＸＭＬ文書のＸＭＬ－ＤＯＭツリーを作成する。

【０１３７】

この場合、上記入力ＵＲＬが、サーバＡ２のウェブ文書Ｗ２を示すＸＰｏｉｎｔｅｒ付ＵＲＬであるので、ＸＰｏｉｎｔｅｒプロセッサ１１５が、ＸＰｏｉｎｔｅｒフラグメント、すなわち、「＃ｘｐｏｉｎｔｅｒ（／／ｔｅｘｔｂｏｏｋ）」を取り出し、ステップＳ２１３で作成されたＸＭＬ－ＤＯＭツリーから当該ＸＰｏｉｎｔｅｒが指し示す「ｔｅｘｔｂｏｏｋ」エレメント（その子孫エレメントを含む部分文書）のＸＭＬ－ＤＯＭツリーを切り出す。「ｔｅｘｔｂｏｏｋ」エレメントが複数ある場合は、それぞれに対して行う。この切り出されたＸＭＬ－ＤＯＭツリーが挿入すべき部分文書のＸＭＬ－ＤＯＭツリーである（ステップＳ２１４）。

【０１３８】

次に、解釈バッファイニシャライザ１１６により、一時解釈バッファ１０４を初期化し、この部分文書にｐｚ：ｔａｒｇｅｔｓエレメントや、ｐｚ：ｃｏｎｖｅｒｔエレメントが記述されているときは、それらの解釈処理を行って、当該部分文書のＸＭＬ文書を得る。

【０１３９】

記述されていないときは、そのまま一時解釈バッファ１０４の解釈処理を終了し、コンテクストマネージャ１２１は、ＤＯＭパーサ１５１を用いて、当該部分文書のＸＭＬ－ＤＯＭツリーからＸＭＬ文書を生成し（ステップＳ２２１）、ｔａｒｇｅｔｓコマンドプロセッサ１２２は、ＤＯＭパーサ１５２を用いて、当該部分文書のＸＭＬ文書のＸＭＬ－ＤＯＭツリーを作成して、これを部分文書群Ｅ２´として、デフォルト解釈バッファ１０３の解釈用ＸＭＬ－ＤＯＭツリー１３１のカレントエレメントであるｐｚ：ｔａｒｇｅｔｓエレメントＥ２と入れ替える。その結果、図１８に示すように、この部分文書群Ｅ２´が、ｐｚ：ｃｏｎｖｅｒｔエレメントＥ３の子エレメントとなり、ＸＭＬ－ＤＯＭツリーが更新される。生成した一時解釈バッファ１０４は破棄する（ステップＳ２２２）。その後、図１３のステップＳ２０８へ戻る。

【０１４０】

図１８に示すように、「ｈｔｔｐ：／／ｗｗｗ．ｘｘｘ．ｃｏｍ／ｂｏｏｋｌｉｓｔ．ｘｍｌ」のウェブ文書内には複数の教科書データが存在するので、その全てが当該ウェブ文書の部分文書のＸＭＬ－ＤＯＭツリーとして挿入されている。

【０１４１】

一方、ステップＳ２０７では、ｐｚ：ｃｏｎｖｅｒｔエレメントＥ３の解釈処理を行うわけだが、ここでの処理動作を図１５に示す。

【０１４２】

ｃｏｎｖｅｒｔコマンドプロセッサ１２３は、ｐｚ：ｃｏｎｖｅｒｔエレメントＥ３のｈｒｅｆ属性値、すなわち、ＸＳＬＴ文書へのＵＲＬ、「ｔｅｘｔｂｏｏｋ－ｂｏｏｋ．ｘｓｌ」取り出し、その属性値を解釈バッファファクトリ１０１の入力ＵＲＬとする。以下のステップＳ２３２～ステップＳ２４０は、ＸＬＭ

文書としてのＸＳＬＴ文書を得るための処理であって、図１４のステップＳ２１２～ステップＳ２２０と同様にして、図１５のステップＳ２４１にて、図１９に示したようなXML文書としてのＸＳＬＴ文書を得る。

【０１４３】

図１９に示すＸＳＬＴ文書は、現在の部分文書の「ｐｕｂｌｉｃａｔｉｏｎ」エレメント、「ｐｒｉｃｅ」エレメント、「ａｕｔｈｏｒ」エレメントを、それぞれ「ｔｉｔｌｅ」エレメント、「ｐｒｉｃｅ」エレメント、「ａｕｔｈｏｒ」エレメントへ変換するための変換ルールを記述したものである。

【０１４４】

図１９に示したようなＸＳＬＴ文書を用いて、ＸＳＬＴプロセッサ１２４は、デフォルト解釈バッファ１０３の解釈用ＸＭＬ－ＤＯＭツリー１３１のカレントエレメントである、ｐｚ：ｃｏｎｖｅｒｔエレメントに含まれる部分文書（子部分文書とも呼ぶ）のＸＭＬ－ＤＯＭツリー上の各子エレメントを変換する（ステップＳ２４２）。

【０１４５】

ここでは、自文書内に含まれている教科書データと、「ｈｔｔｐ：／／ｗｗｗ．ｘｘｘ．ｃｏｍ／ｂｏｏｋｌｉｓｔ．ｘｍｌ」のウェブ文書から抽出した教科書データは同じ構造のデータであるので、エレメントＥ１の自文書内含まれていた教科書データの場合を例にとり、図１９のＸＳＬＴ文書を用いて、その構造を変換する場合を説明する。

【０１４６】

図１６に示すように、エレメントＥ１の子エレメントである「ｐｕｂｌｉｃａｔｉｏｎ」エレメントの値は、「Ｓｅｌｅｃｔｅｄ　Ｓｈｏｒｔ　Ｓｔｏｒｉｅｓ　ｏｆ　Ｓｈｉｎｉｃｈｉｒｏ　Ｈａｍａｄａ」であるが、これは、変換後では、「ｔｉｔｌｅ」エレメントの値となる。また、図１６において、エレメントＥ１の子エレメントである「ａｕｔｈｏｒ」エレメントの値は「Ｓｈｉｎｉｃｈｉｒｏ　Ｈａｍａｄａ」であるが、これは変換後では、「ａｕｔｈｏｒ」エレメントとなる。さらに、図１６に示すように、エレメントＥ１の子エレメントである「ｐｒｉｃｅ」エレメントの値は、「５５」であるが、これは変換後も同じで

ある。

【０１４７】

ｃｏｎｖｅｒｔコマンドプロセッサ１２３は、変換後の部分文書のＸＭＬ－ＤＯＭツリーを、新たなエレメントＥ３′として、デフォルト解釈バッファ１０３の解釈用ＸＭＬ－ＤＯＭツリー１３１のカレントエレメントであるｐｚ：ｃｏｎｖｅｒｔエレメントＥ３と入れ替えて、図２０に示したような文書構造のＸＭＬ－ＤＯＭツリーが生成される。

【０１４８】

なお、生成した一時解釈バッファ１０４は破棄する（ステップＳ２４３）。その後、図１３のステップＳ２０８へ戻る。

【０１４９】

以上のようにして、デフォルト解釈バッファ１０３のプログラムカウンタ１３２が「ＮＵＬＬ」となり、ＸＭＬ－ＤＯＭツリー１３１の解釈が終了すると、コンテクストマネージャ１２１は、ＸＭＬ－ＤＯＭパーサ１５１を用いて、図２０に示したＸＭＬ－ＤＯＭツリーを含む解釈バッファ１０３のＸＭＬ－ＤＯＭツリー１３１を基に、目的とするウェブ文書Ｗ１としてのＸＭＬ文書を生成し出力する。

【０１５０】

なお、クライアント端末Ｂ１のウェブブラウザがＸＭＬ文書を表示できる場合は、ＸＭＬ文書のウェブ文書Ｗ１をそのままクライアント端末Ｂ１のウェブブラウザに返すが、表示できない場合は、サーバＡ１側でスタイルシートを処理して、ウェブ文書Ｗ１をＨＴＭＬ文書に変換してからクライアント端末Ｂ１のウェブブラウザへ返す（図１３のステップＳ２０９）。

【０１５１】

（Ｄ）ウェブ文書の合成処理のためのＸＭＬ－Ｐ’ｚサーバ間の協調動作

次に、ウェブ文書の合成処理をＸＭＬ－Ｐ’ｚサーバ間で協調して行う場合について説明する。

【０１５２】

例えば、あるＸＭＬ－Ｐ’ｚサーバ上のＸＭＬ－Ｐ’ｚ文書を解釈処理中に他のＸＭＬ－Ｐ’ｚサーバのＸＭＬ－Ｐ’ｚ文書を挿入する場合に、その挿入されるＸＭＬ－Ｐ’ｚ文書は、どちらのサーバが解釈するのかという問題がある。すなわち、ＧＥＴコマンドによる要求があった場合に、ＸＭＬ－Ｐ’ｚ文書そのものを返すのか、解釈処理した結果のＸＭＬ文書を返すのかという判断を行う必要があるということである。

【０１５３】

ＨＴＴＰサーバ（ＸＭＬ－Ｐ’ｚ文書を要求される側）とＨＴＴＰクライアント（ＸＭＬ－Ｐ’ｚ文書を要求する側）との間で、ＨＴＴＰクライアントがＸＭＬ－Ｐ’ｚ文書を解釈処理できない場合は、ＨＴＴＰサーバ側でＸＭＬ－Ｐ’ｚ文書を解釈処理しなければならないという制約がある。

【０１５４】

この制約を判断の材料に導入するため、ＸＭＬ－Ｐ’ｚ言語処理系１００の解釈バッファファクトリ１０１が、ＸＭＬ－Ｐ’ｚ文書を要求する際に、ＧＥＴコマンドによる要求のヘッダに「ＸＭＬ－Ｐ’ｚ：　ｅｎａｂｌｅ」をつけるものとする。

【０１５５】

また、ＨＴＴＰサーバとしては、ＸＭＬ－Ｐ’ｚ文書の解釈処理をＨＴＴＰクライアントに委譲することにより、サーバの負荷を下げることができる利点もあるが、ＸＭＬ－Ｐ’ｚ文書を公開したくない何らかの理由があるかもしれない（含まれている合成ロジックを公開したくないなど）ので、サーバ側でＸＭＬ－Ｐ’ｚ言語を解釈処理するかどうかは設定次第である。

【０１５６】

以上を踏まえて、ＨＴＴＰサーバが解釈実行するかどうかの判断処理動作について、図１０の示すフローチャートを参照して説明する。

【０１５７】

まず、ステップＳ７１では、ＧＥＴ要求のヘッダに「ＸＭＬ－Ｐ’ｚ：ｅｎａｂｌｅ」が含まれているかどうかを調べ、含まれていないならば、ステップＳ７２へ進み、ＨＴＴＰサーバ上でＸＭＬ－Ｐ’ｚ文書を解釈処理して終了する。含

まれているならば、ステップＳ７３へ進み、ＨＴＴＰサーバがＸＭＬ－Ｐ’ｚ文書を処理する設定になっているかどうかをチェックし、そうであれば、ステップＳ７４へ進み、ＨＴＴＰサーバでＸＭＬ－Ｐ’ｚ文書を解釈処理して終了し、そうでなければ、ステップＳ７５へ進み、解釈処理をしないでＨＴＴＰクライアントにＸＭＬ－Ｐ’ｚ文書をそのまま送信して終了する。

【０１５８】

（Ｅ）追記

以上説明したように、上記実施形態によれば、合成のためのベースとなる合成用ウェブ文書をＸＭＬで記述し、指定した他のウェブ文書から指定した範囲の部分（部分文書）を抽出して、それを合成用ウェブ文書の指定された位置に挿入し、合成用ウェブ文書の指定した範囲に変換処理を施す、挿入・変換の２つの合成ロジック命令をその合成用ウェブ文書内にエレメントとして持たせたＸＭＬ－Ｐ’ｚ（ＸＭＬ－Ｐｉｅｃｅｓ）文書を定義する。言語処理系１００は、ＸＭＬ－Ｐ’ｚ文書に記述されている、指定されたウェブサーバ（例えば、ここでは、ウェブサーバＡ２、Ａ３）のウェブ文書（ページ）Ｗ２、Ｗ３から指定した範囲の部分（部分文書）を抽出し、それをＸＭＬ－Ｐ’ｚ文書の指定位置に挿入するとともに、ＸＭＬ－Ｐ’ｚ文書に記述されている指定された範囲に変換処理を施す。最終的に、ＸＭＬ－Ｐ’ｚ言語処理系１００の処理結果としてのＸＭＬ文書（合成されたウェブ文書）Ｗ１を得ることにより、複数のウェブサイトの情報を１つのウェブ文書上に合成することが容易にしかも汎用的に行える。

【０１５９】

なお、上記実施形態に記載した手法は、コンピュータに実行させることのできるプログラムとして、ＤＶＤ、ＣＤ－ＲＯＭ、フロッピディスク、個体メモリ、光ディスクなどの記録媒体に格納して頒布することもできる。

【０１６０】

【発明の効果】

以上説明したように、本発明によれば、複数のウェブサイトの情報を１つのウェブ文書上に合成することが容易にしかも汎用的に行える。

【図面の簡単な説明】

【図１】

本発明のＸＭＬ－Ｐ’ｚ言語処理系を組み込んだウェブサーバ（ＸＭＬ－Ｐ’ｚサーバ）の基本的な動作を説明するための図。

【図２】

ＸＭＬ－Ｐ’ｚ言語処理系の全体の構成例を示した図。

【図３】

ＨＴＭＬ判定器において、与えられたＵＲＬにて指定されるウェブ文書がＨＴＭＬ文書かＸＭＬ文書かを判定するための処理動作を示したフローチャート。

【図４】

ＨＴＭＬ－ＸＭＬコンバータのＨＴＭＬ文書からＸＭＬ文書への変換処理動作を説明するためのフローチャート。

【図５】

ＸＰｏｉｎｔｅｒプロセッサのＸＰｏｉｎｔｅｒフラグメントに対する処理動作を説明するためのフローチャート。

【図６】

解釈バッファイニシャライザの解釈バッファの初期化処理動作を説明するためのフローチャート。

【図７】

コンテクストマネージャの処理動作を説明するためのフローチャート。

【図８】

ｔａｒｇｅｔｓコマンドプロセッサのｔａｒｇｅｔｓエレメントの解釈処理動作を説明するためのフローチャート。

【図９】

ｃｏｎｖｅｒｔコマンドプロセッサのｃｏｎｖｅｒｔエレメントの解釈処理動作を説明するためのフローチャート。

【図１０】

ＸＭＬ－Ｐ’ｚ文書の解釈処理をサーバ側で行うかクライアント側で行うかを判断する判断処理動作について説明するためのフローチャート。、

【図１１】

（ａ）図は、ＸＭＬ－Ｐ’ｚ文書の第１の例の文書構造を模式的に示した図で、（ｂ）図は、ＸＭＬ－Ｐ’ｚ文書の解釈後のＸＭＬ文書の文書構造を示した図。

【図１２】

ＸＭＬ－Ｐ‘ｚ文書の解釈順序について説明するための図。

【図１３】

図２に示した構成の言語処理系が、複数のウェブ文書を１つのウェブ文書上に合成するための連の動作を説明するためのフローチャート。

【図１４】

図２に示した構成の言語処理系が、複数のウェブ文書を１つのウェブ文書上に合成するための連の動作を説明するためのフローチャート。

【図１５】

図２に示した構成の言語処理系が、複数のウェブ文書を１つのウェブ文書上に合成するための連の動作を説明するためのフローチャート。

【図１６】

合成用ウェブ文書としてのＸＭＬ－Ｐ‘ｚ文書の一例であって、ＸＭＬ－Ｐ‘ｚ文書の一部を示した図。

【図１７】

図１６のＸＭＬ－Ｐ‘ｚ文書に対応するＸＭＬ－ＤＯＭツリーを概略的に示した図。

【図１８】

図１６のｐｚ：ｔａｒｇｅｔｓエレメントを解釈した結果のＸＭＬ－ＤＯＭツリーを概略的に示した図。

【図１９】

図１６のＸＭＬ－Ｐ‘ｚ文書に記述されているＸＳＬＴ文書の一例を示した図。

【図２０】

図１６のｐｚ：ｔａｒｇｅｔｓエレメントとｐｚ：ｃｏｎｖｅｒｔエレメント

を解釈した結果のＸＭＬ－ＤＯＭツリーを概略的に示した図。

【符号の説明】

Ａ１、Ａ２、Ａ３…サーバ

Ｂ１…クライアント端末

Ｗ１…合成されたウェブ文書（ＸＭＬ文書）

Ｗ２～Ｗ３…ウェブ文書

１…ＸＭＬ－Ｐ’ｚ言語処理系（合成処理部）

２…ＸＭＬ－Ｐ’ｚ文書

１００…ＸＭＬ－Ｐ’ｚ言語処理系

１０１…解釈バッファファクトリ

１０２…インタプリタ

１０３…デフォルト解釈バッファ

１０４…一時解釈バッファ

１１１…ＸＭＬノーマライザ

１１２…ＨＴＭＬ判定器

１１３…ＨＴＭＬ－ＸＭＬコンバータ

１１４…ＸＭＬ－ＤＯＭパーサ

１１５…ＸＰｏｉｎｔｅｒプロセッサ

１１６…解釈バッファイニシャライザ

１２１…コンテクストマネージャ

１２２…ｔａｒｇｅｔｓコマンドマネージャ

１２３…ｃｏｎｖｅｒｔコマンドマネージャ

１２４…ＸＳＬＴプロセッサ

１３１…解釈用ＸＭＬ－ＤＯＭツリー

１３２…プログラムカウンタ

１３３…ロードフラグ

１３４…ソースＸＭＬ－ＤＯＭツリー

１４１…解釈用ＸＭＬ－ＤＯＭツリー

１４２…プログラムカウンタ

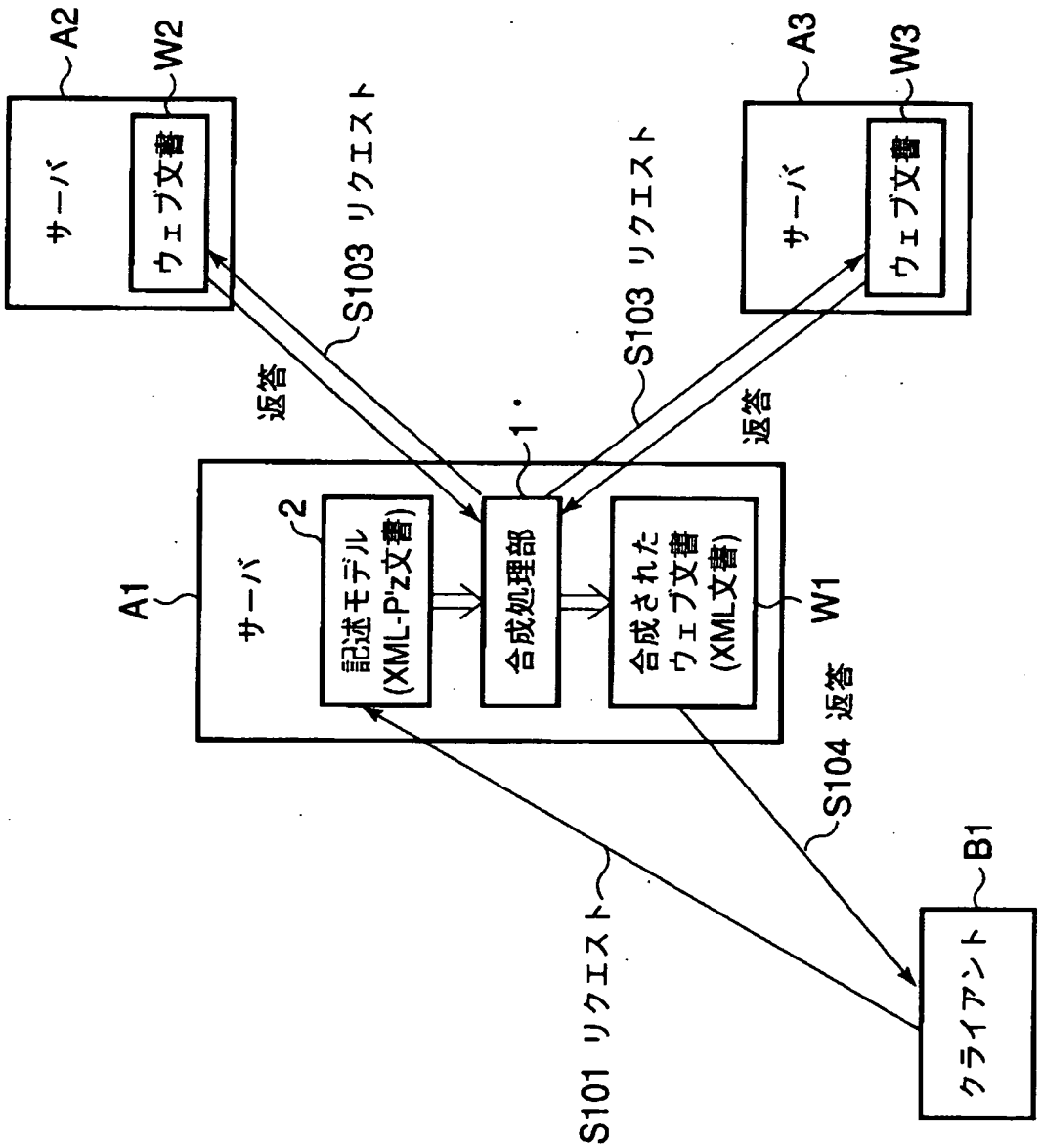
１４３…ロードフラグ

１４４…ソースＸＭＬ－ＤＯＭツリー

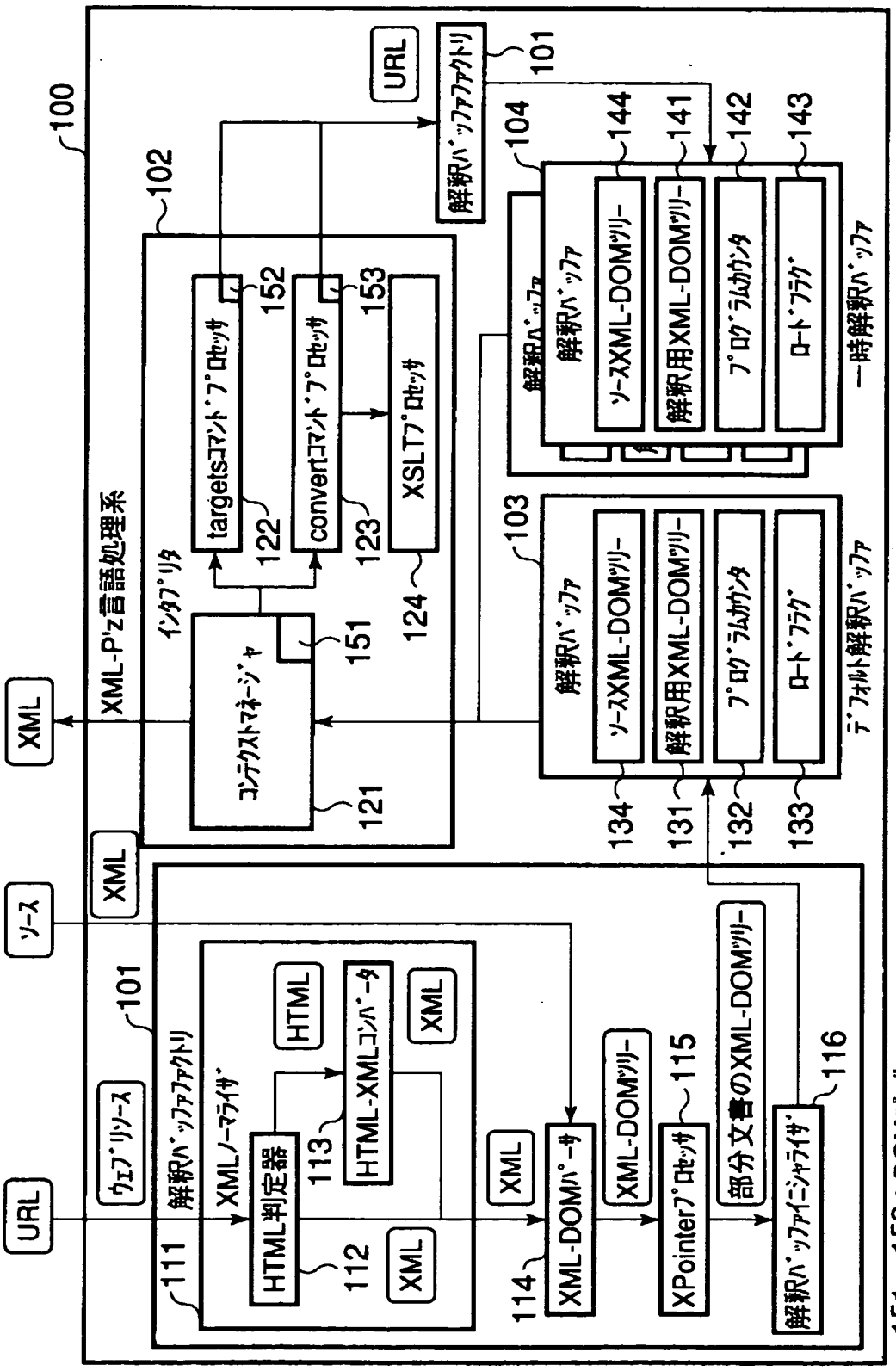
１５１～１５３…ＤＯＭパーサ

【書類名】　図面

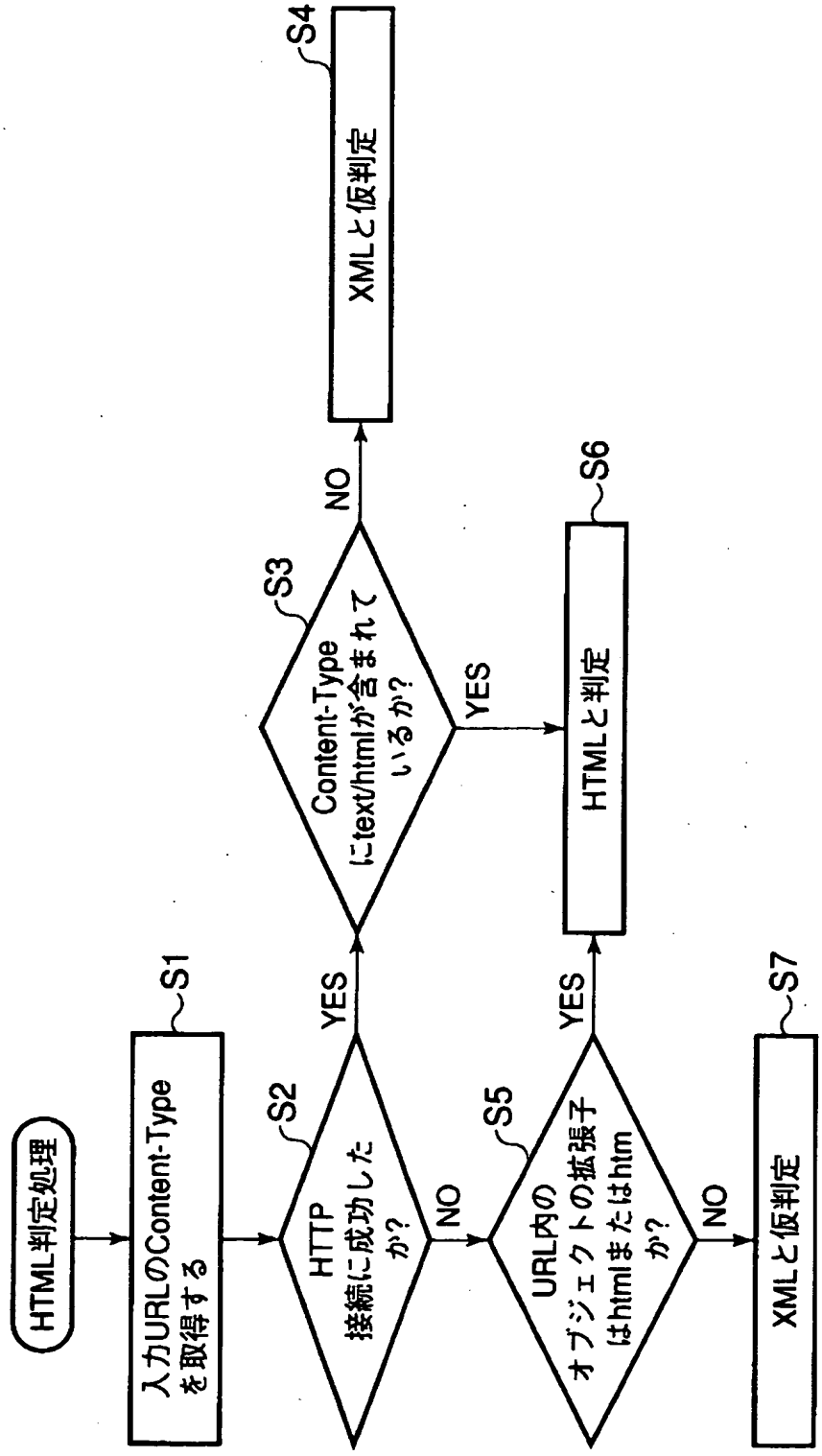
【図１】


A2
W2
サーバ
ウェブ文書
S103 リクエスト
返答
A3
W3
サーバ
ウェブ文書
S103 リクエスト
返答
1
A1
サーバ
2
記述モデル
(XML-P'z文書)
合成処理部
合成された
ウェブ文書
(XML文書)
W1
S104 返答
S101 リクエスト
B1
クライアント

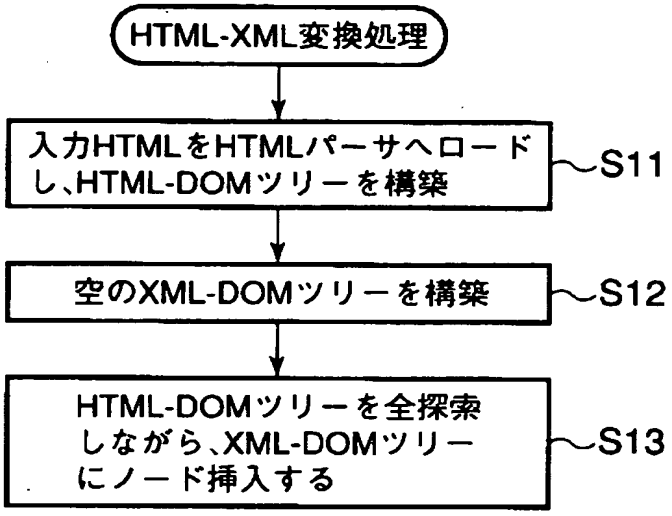
【図２】


100
XML-P'z言語処理系
インタプリタ
102
targetsコマンドプログラム
152
convertコマンドプログラム
153
XSLTプログラム
122
123
124
コンテクストマネージャ
151
121
URL
解釈バッファファクトリ
101
104
解釈バッファ
ソースXML-DOMツリー
144
解釈用XML-DOMツリー
141
プログラムカウンタ
142
ロードプログラム
143
一時解釈バッファ
103
デフォルト解釈バッファ
134
131
132
133
XML
ソース
URL
ウェブリソース
解釈バッファファクトリ
101
111
XMLノーマライザ
HTML判定器
112
HTML-XMLコンバータ
113
HTML
XML
XML
XML-DOMパーサ
114
XML-DOMツリー
XPointerプログラム
115
部分文書のXML-DOMツリー
解釈バッファイニシャライザ
116
151～153：DOMパーサ

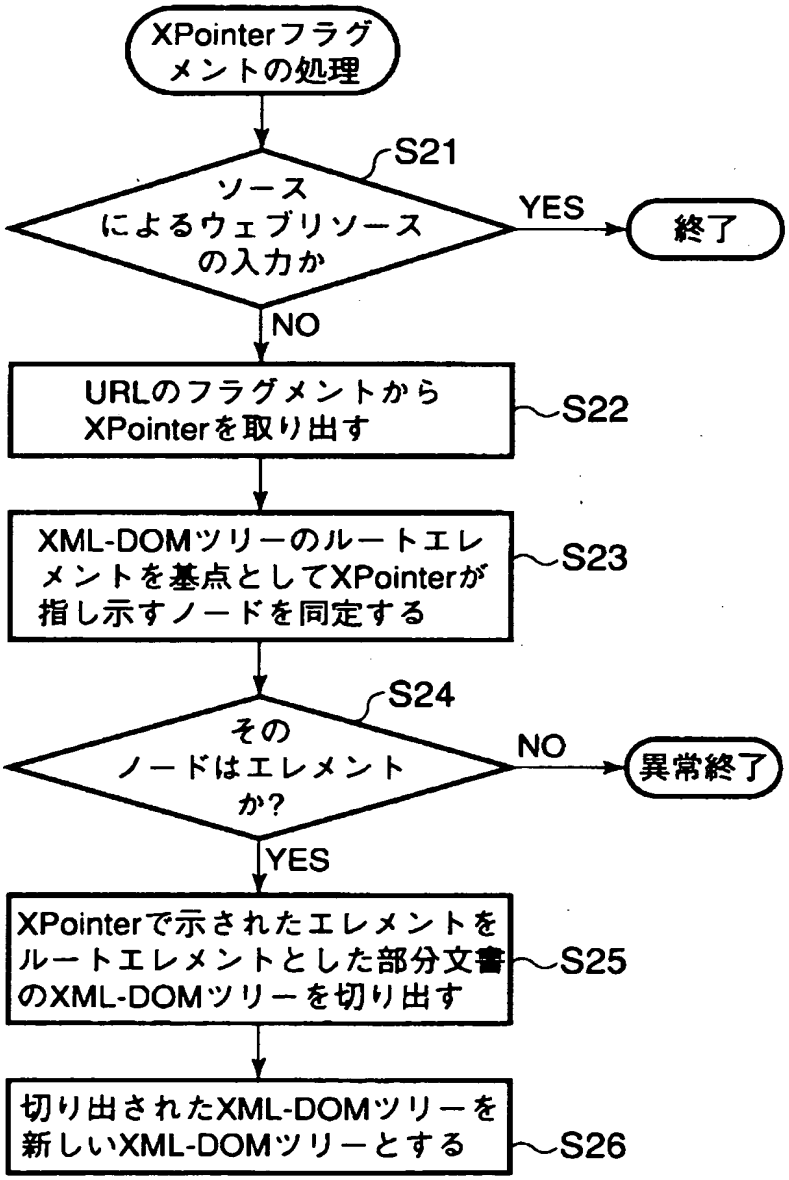
【図３】


HTML判定処理
S1
入力URLのContent-Typeを取得する
S2
HTTP接続に成功したか？
YES
NO
S3
Content-Typeにtext/htmlが含まれているか？
YES
NO
S4
XMLと仮判定
S5
URL内のオブジェクトの拡張子はhtmlまたはhtmか？
YES
NO
S6
HTMLと判定
S7
XMLと仮判定

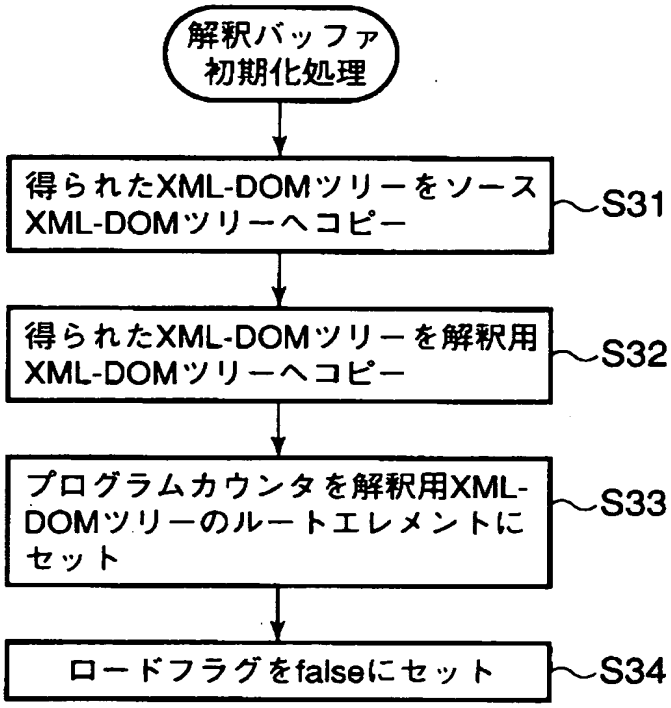
【図４】


HTML-XML変換処理
入力HTMLをHTMLパーサへロードし、HTML-DOMツリーを構築
S11
空のXML-DOMツリーを構築
S12
HTML-DOMツリーを全探索しながら、XML-DOMツリーにノード挿入する
S13

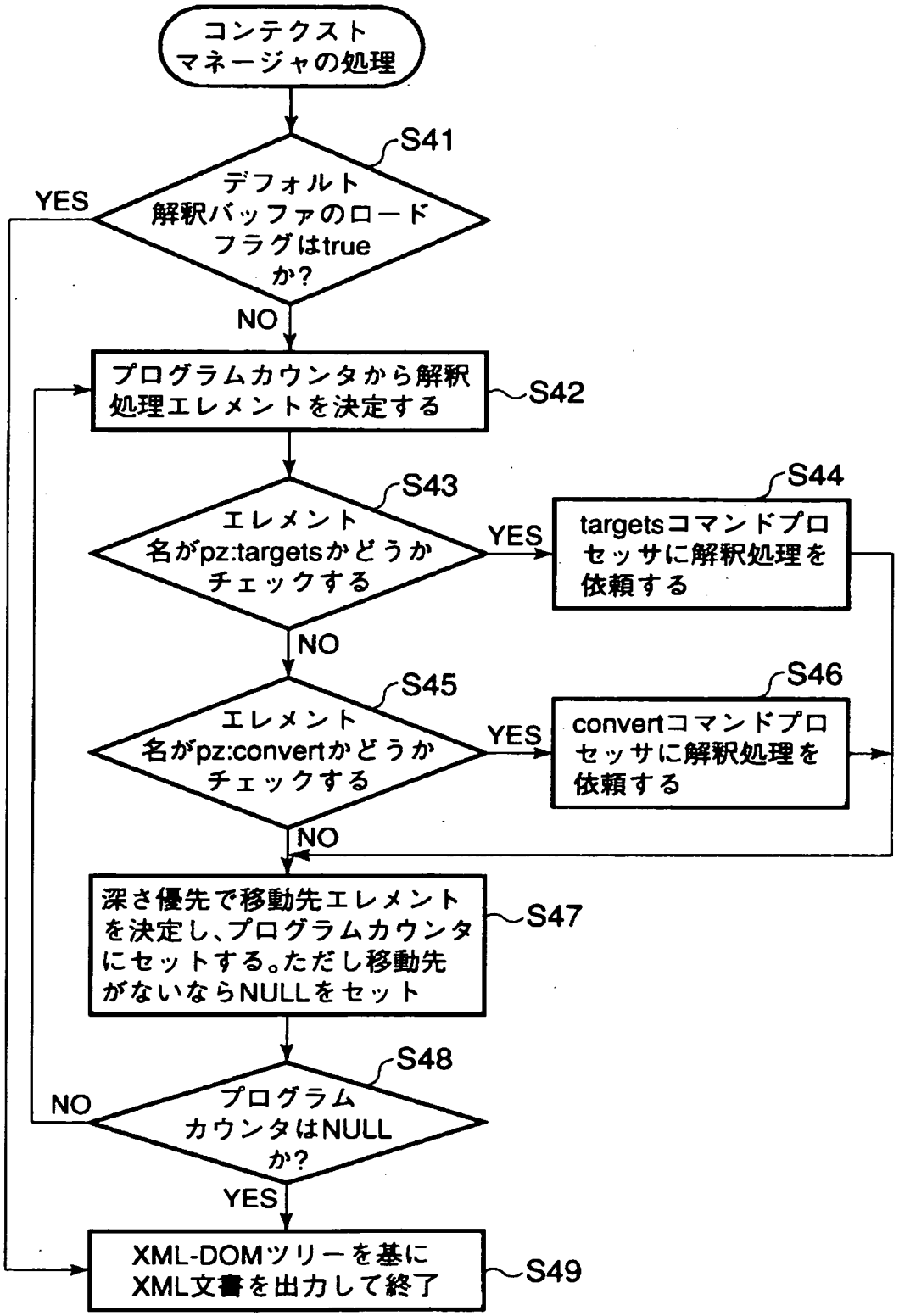
【図５】


XPointerフラグメントの処理
S21
ソースによるウェブリソースの入力か
YES
終了
NO
URLのフラグメントからXPointerを取り出す
S22
XML-DOMツリーのルートエレメントを基点としてXPointerが指し示すノードを同定する
S23
S24
そのノードはエレメントか?
NO
異常終了
YES
XPointerで示されたエレメントをルートエレメントとした部分文書のXML-DOMツリーを切り出す
S25
切り出されたXML-DOMツリーを新しいXML-DOMツリーとする
S26

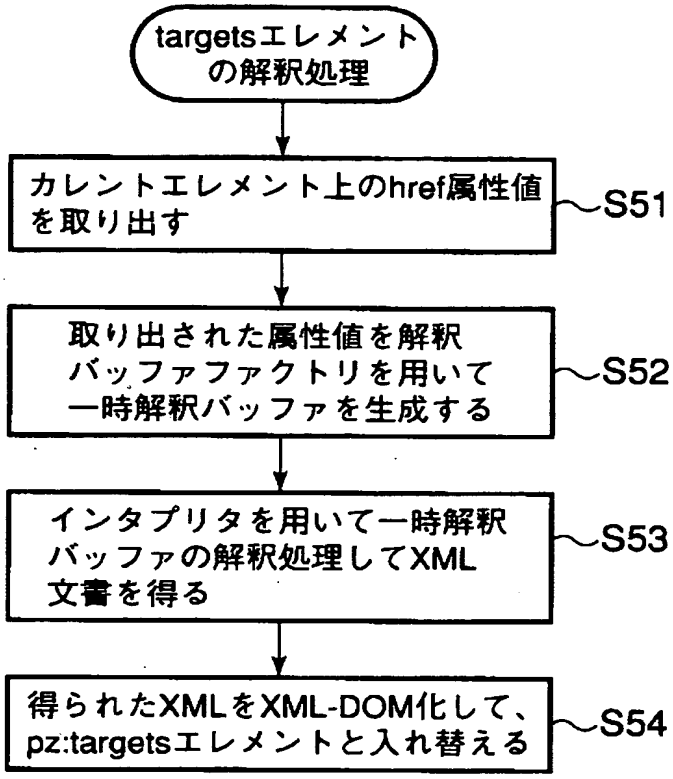
【図６】


解釈バッファ
初期化処理
得られたXML-DOMツリーをソース
XML-DOMツリーへコピー
S31
得られたXML-DOMツリーを解釈用
XML-DOMツリーへコピー
S32
プログラムカウンタを解釈用XML-
DOMツリーのルートエレメントに
セット
S33
ロードフラグをfalseにセット
S34

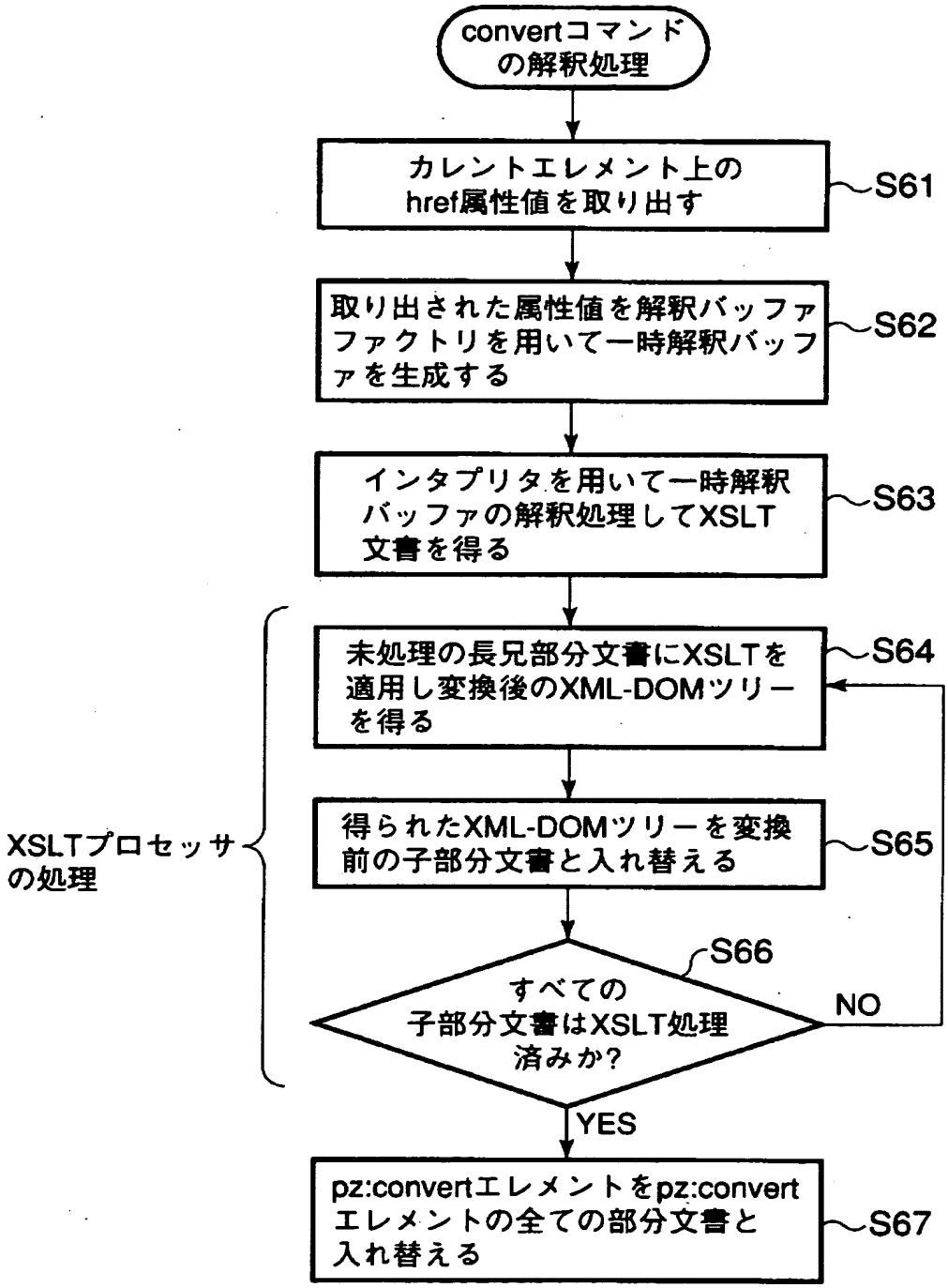
【図７】


コンテクスト
マネージャの処理
S41
デフォルト
解釈バッファのロード
フラグはtrue
か?
YES
NO
プログラムカウンタから解釈
処理エレメントを決定する
S42
S43
エレメント
名がpz:targetsかどうか
チェックする
YES
S44
targetsコマンドプロ
セッサに解釈処理を
依頼する
NO
S45
エレメント
名がpz:convertかどうか
チェックする
YES
S46
convertコマンドプロ
セッサに解釈処理を
依頼する
NO
深さ優先で移動先エレメント
を決定し、プログラムカウンタ
にセットする。ただし移動先
がないならNULLをセット
S47
S48
プログラム
カウンタはNULL
か?
NO
YES
XML-DOMツリーを基に
XML文書を出力して終了
S49

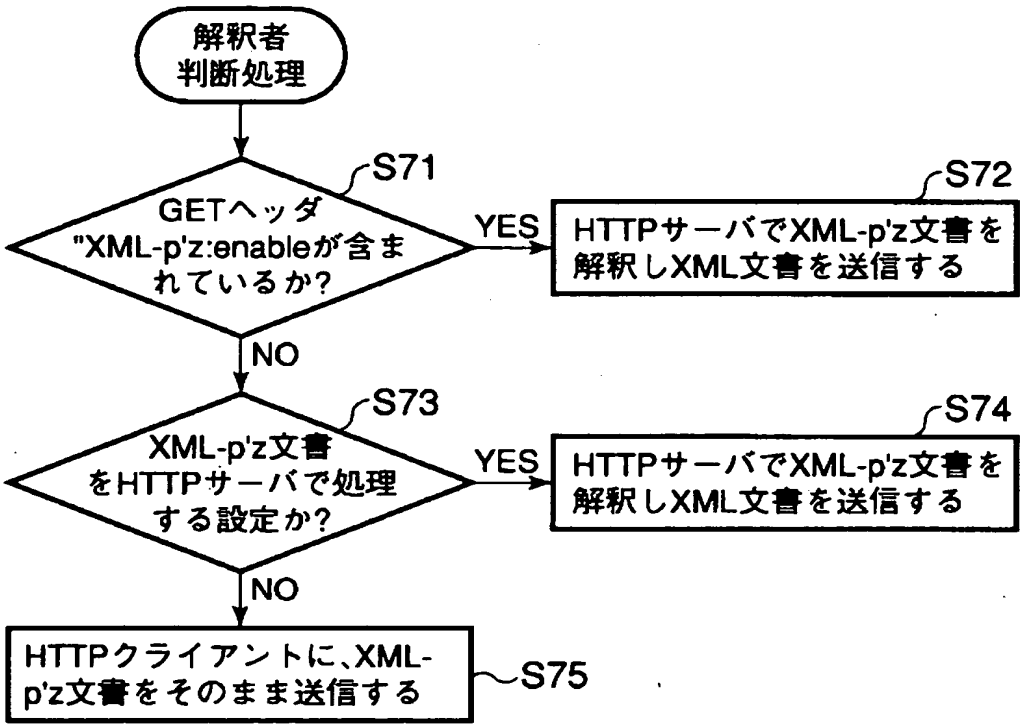
【図８】


targetsエレメント
の解釈処理
カレントエレメント上のhref属性値
を取り出す
S51
取り出された属性値を解釈
バッファファクトリを用いて
一時解釈バッファを生成する
S52
インタプリタを用いて一時解釈
バッファの解釈処理してXML
文書を得る
S53
得られたXMLをXML-DOM化して、
pz:targetsエレメントと入れ替える
S54

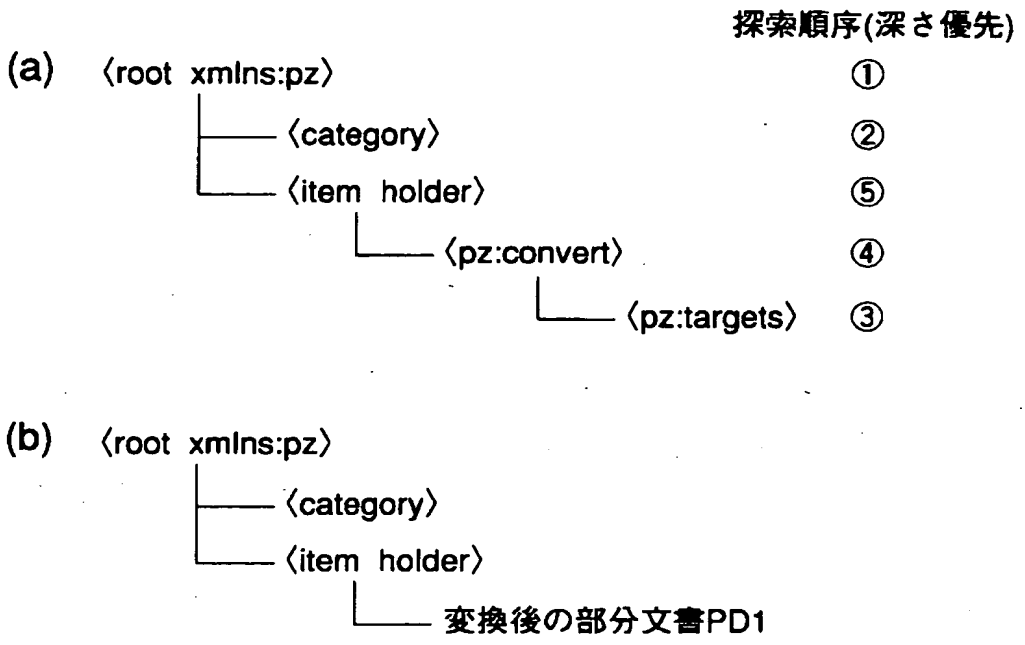
【図９】


convertコマンド
の解釈処理
カレントエレメント上の
href属性値を取り出す
S61
取り出された属性値を解釈バッファ
ファクトリを用いて一時解釈バッフ
ァを生成する
S62
インタプリタを用いて一時解釈
バッファの解釈処理してXSLT
文書を得る
S63
未処理の長兄部分文書にXSLTを
適用し変換後のXML-DOMツリー
を得る
S64
得られたXML-DOMツリーを変換
前の子部分文書と入れ替える
S65
S66
すべての
子部分文書はXSLT処理
済みか?
NO
YES
pz:convertエレメントをpz:convert
エレメントの全ての部分文書と
入れ替える
S67
XSLTプロセッサ
の処理

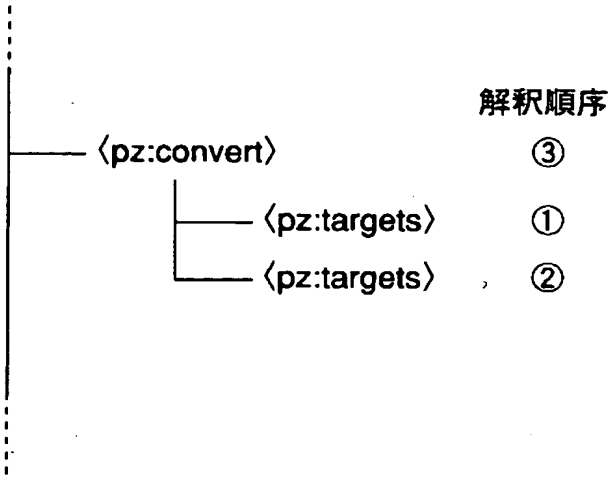
【図１０】


解釈者
判断処理
S71
GETヘッダ
"XML-p'z:enableが含まれているか?
YES
S72
HTTPサーバでXML-p'z文書を解釈しXML文書を送信する
NO
S73
XML-p'z文書をHTTPサーバで処理する設定か?
YES
S74
HTTPサーバでXML-p'z文書を解釈しXML文書を送信する
NO
HTTPクライアントに、XML-p'z文書をそのまま送信する
S75

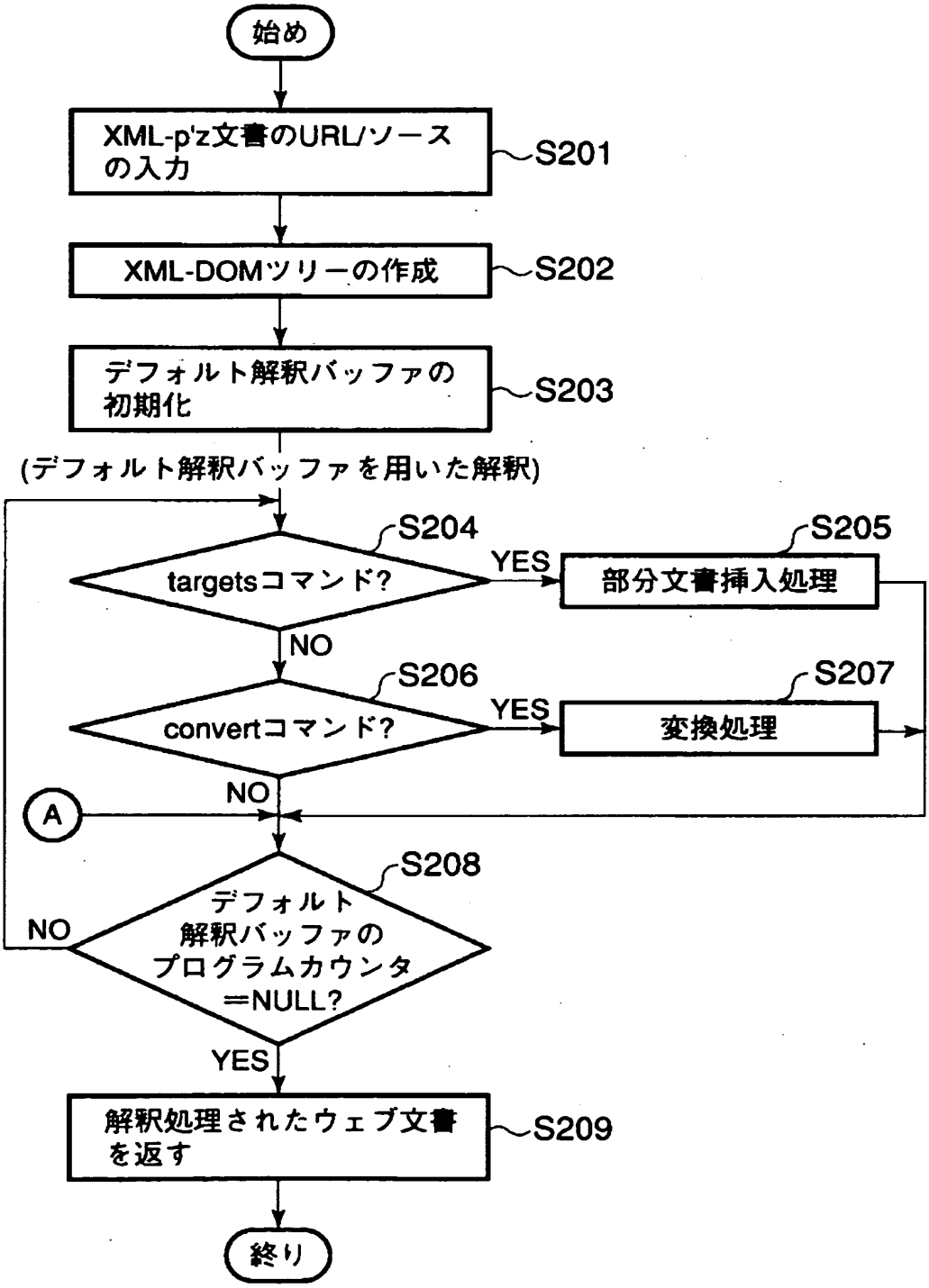
【図１１】


探索順序(深さ優先)
(a)
〈root xmlns:pz〉
①
〈category〉
②
〈item holder〉
⑤
〈pz:convert〉
④
〈pz:targets〉
③
(b)
〈root xmlns:pz〉
〈category〉
〈item holder〉
変換後の部分文書PD1

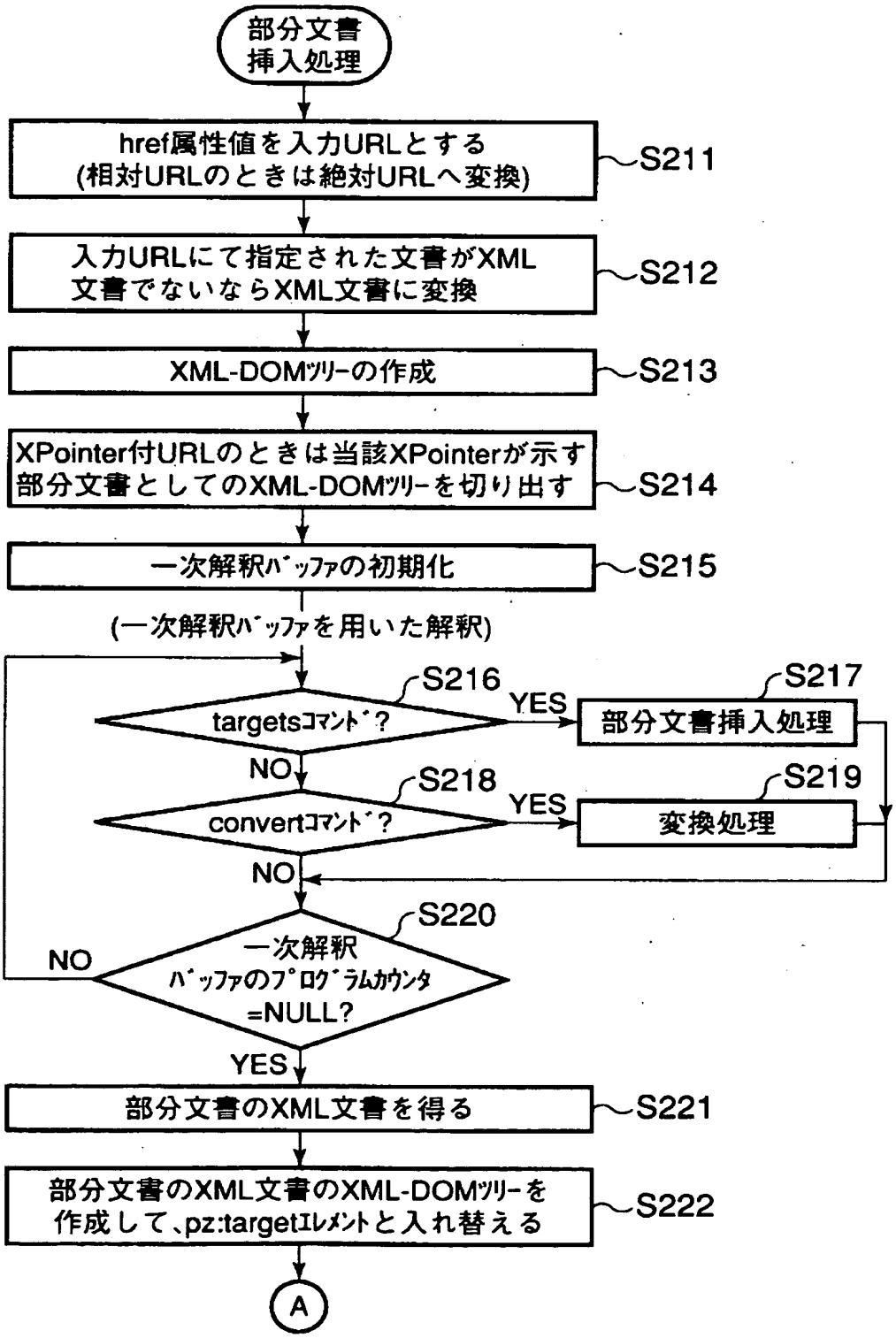
【図１２】



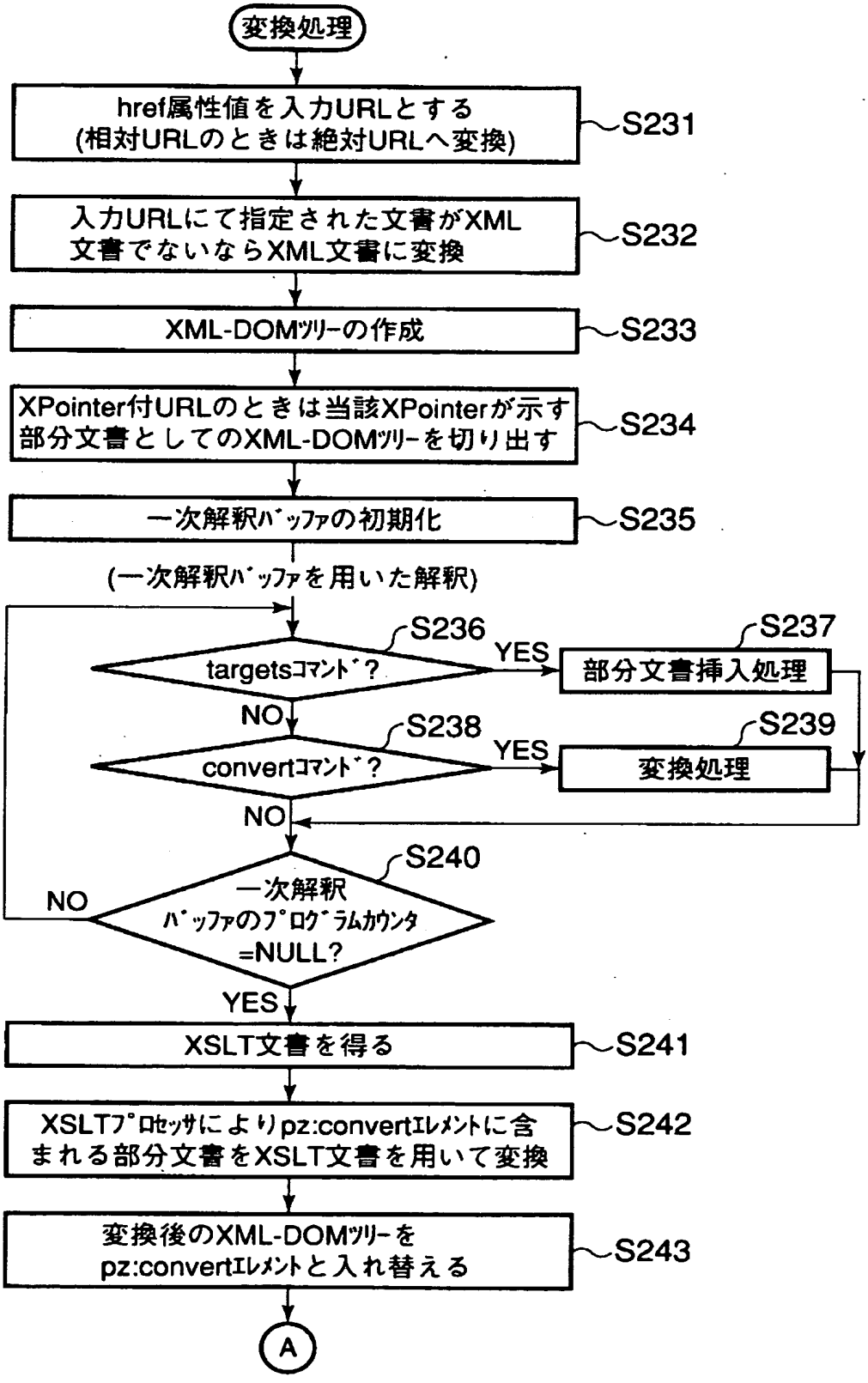
【図１３】


始め
XML-p'z文書のURL/ソースの入力
S201
XML-DOMツリーの作成
S202
デフォルト解釈バッファの初期化
S203
(デフォルト解釈バッファを用いた解釈)
S204
targetsコマンド?
YES
S205
部分文書挿入処理
NO
S206
convertコマンド?
YES
S207
変換処理
NO
A
S208
デフォルト解釈バッファのプログラムカウンタ=NULL?
NO
YES
解釈処理されたウェブ文書を返す
S209
終り

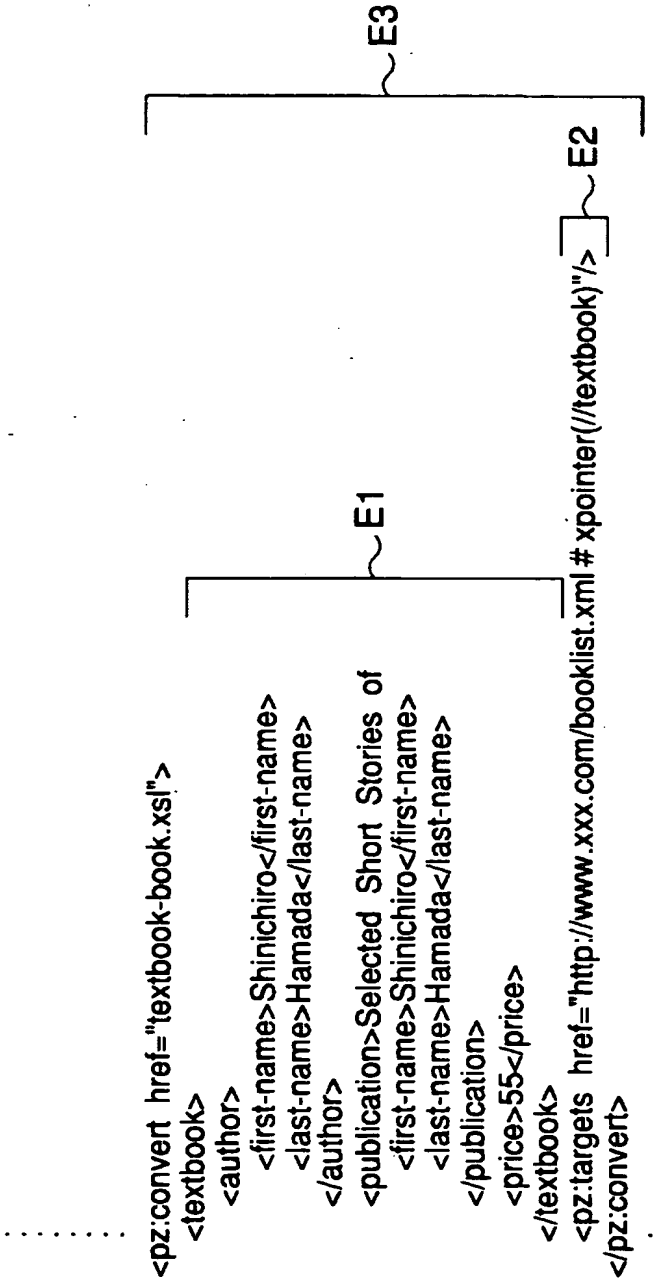
【図１４】


部分文書
挿入処理
href属性値を入力URLとする
(相対URLのときは絶対URLへ変換)
S211
入力URLにて指定された文書がXML
文書でないならXML文書に変換
S212
XML-DOMﾂﾘｰの作成
S213
XPointer付URLのときは当該XPointerが示す
部分文書としてのXML-DOMﾂﾘｰを切り出す
S214
一次解釈ﾊﾞｯﾌｧの初期化
S215
(一次解釈ﾊﾞｯﾌｧを用いた解釈)
S216
targetsｺﾏﾝﾄﾞ?
YES
S217
部分文書挿入処理
NO
S218
convertｺﾏﾝﾄﾞ?
YES
S219
変換処理
NO
S220
一次解釈
ﾊﾞｯﾌｧのﾌﾟﾛｸﾞﾗﾑｶｳﾝﾀ
=NULL?
NO
YES
部分文書のXML文書を得る
S221
部分文書のXML文書のXML-DOMﾂﾘｰを
作成して、pz:targetｴﾚﾒﾝﾄと入れ替える
S222
A

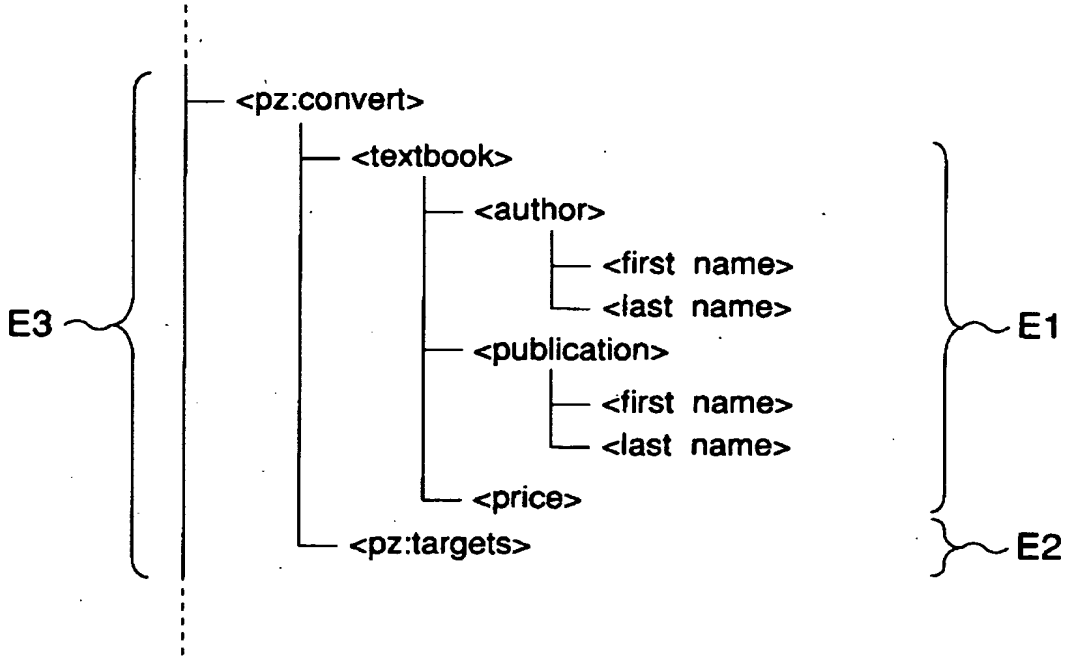
【図１５】


変換処理
href属性値を入力URLとする
(相対URLのときは絶対URLへ変換)
S231
入力URLにて指定された文書がXML
文書でないならXML文書に変換
S232
XML-DOMﾂﾘｰの作成
S233
XPointer付URLのときは当該XPointerが示す
部分文書としてのXML-DOMﾂﾘｰを切り出す
S234
一次解釈ﾊﾞｯﾌｧの初期化
S235
(一次解釈ﾊﾞｯﾌｧを用いた解釈)
S236
targetsｺﾏﾝﾄﾞ?
YES
S237
部分文書挿入処理
NO
S238
convertｺﾏﾝﾄﾞ?
YES
S239
変換処理
NO
S240
一次解釈
ﾊﾞｯﾌｧのﾌﾟﾛｸﾞﾗﾑｶｳﾝﾀ
=NULL?
NO
YES
XSLT文書を得る
S241
XSLTﾌﾟﾛｾｯｻによりpz:convertｴﾚﾒﾝﾄに含
まれる部分文書をXSLT文書を用いて変換
S242
変換後のXML-DOMﾂﾘｰを
pz:convertｴﾚﾒﾝﾄと入れ替える
S243
A

【図１６】

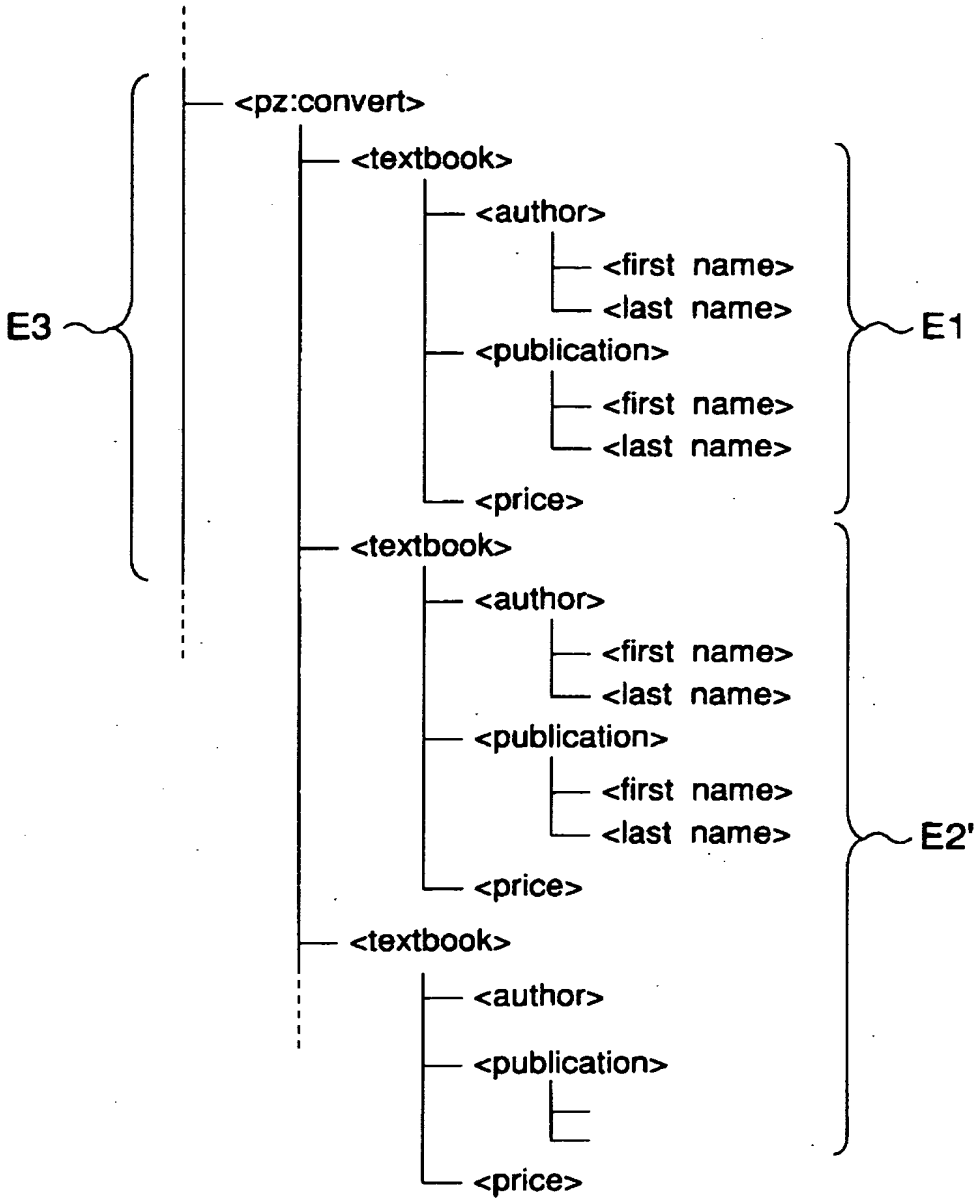

E3
E2
E1
<pz:convert href="textbook-book.xsl">
<textbook>
<author>
<first-name>Shinichiro</first-name>
<last-name>Hamada</last-name>
</author>
<publication>Selected Short Stories of
<first-name>Shinichiro</first-name>
<last-name>Hamada</last-name>
</publication>
<price>55</price>
</textbook>
<pz:targets href="http://www.xxx.com/booklist.xml # xpointer(//textbook)"/>
</pz:convert>

【図１７】

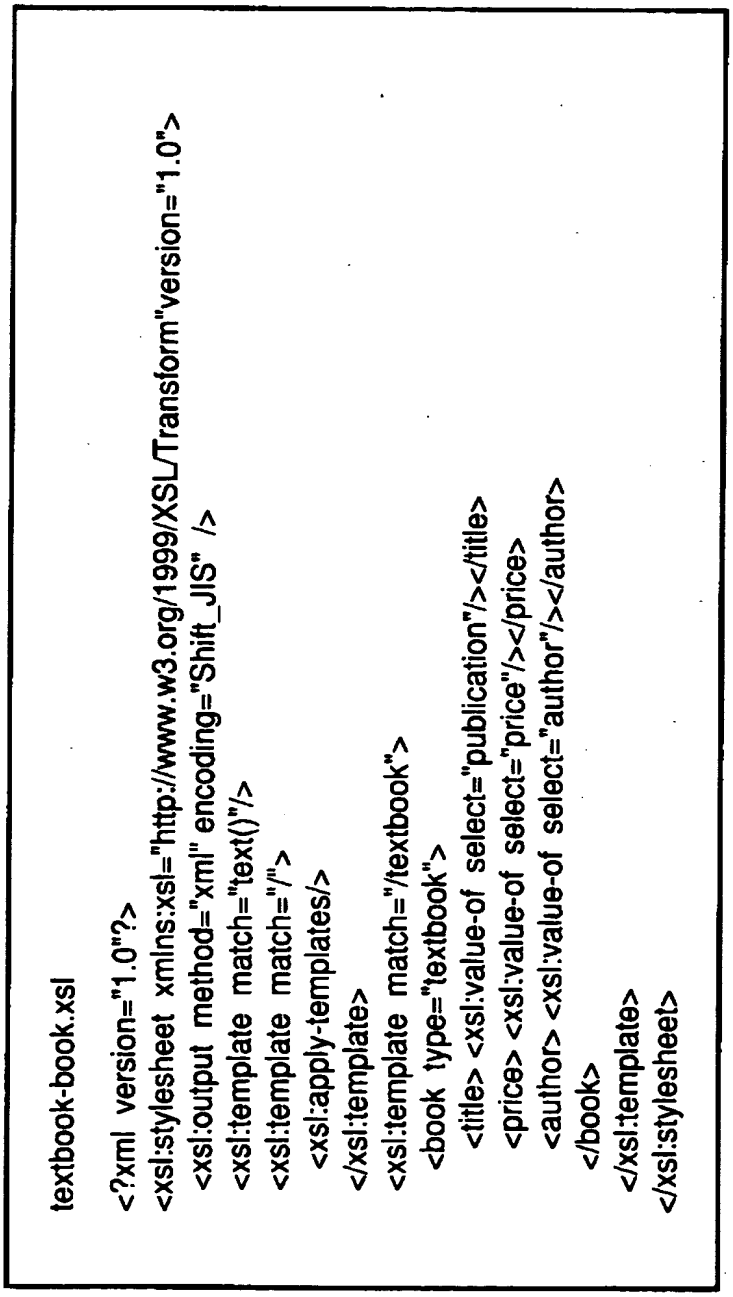


出証特２００１－３０７９８６６

【図１８】


<pz:convert>
<textbook>
<author>
<first name>
<last name>
<publication>
<first name>
<last name>
<price>
<textbook>
<author>
<first name>
<last name>
<publication>
<first name>
<last name>
<price>
<textbook>
<author>
<publication>
<price>
E3
E1
E2'

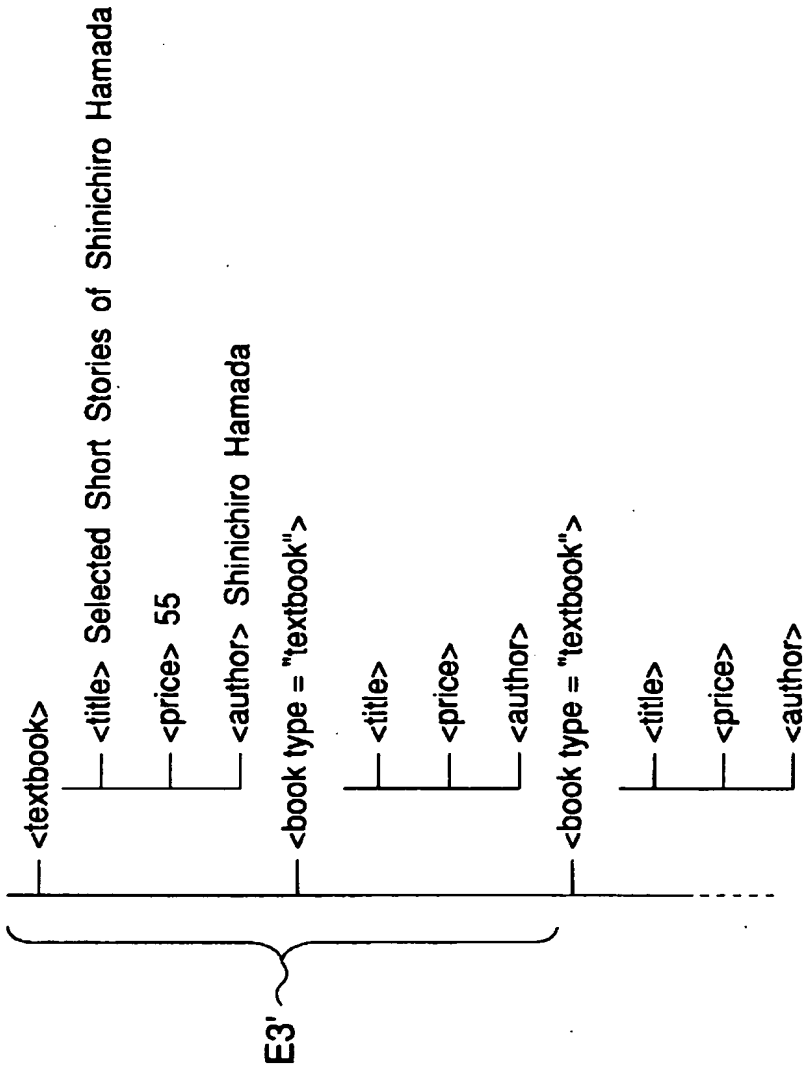
【図１９】



textbook-book.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"version="1.0">
  <xsl:output method="xml" encoding="Shift_JIS" />
  <xsl:template match="text()"/>
  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="/textbook">
    <book type="textbook">
      <title> <xsl:value-of select="publication"/></title>
      <price> <xsl:value-of select="price"/></price>
      <author> <xsl:value-of select="author"/></author>
    </book>
  </xsl:template>
</xsl:stylesheet>
```

【図２０】


<textbook>
<title> Selected Short Stories of Shinichiro Hamada
<price> 55
<author> Shinichiro Hamada
<book type = "textbook">
<title>
<price>
<author>
<book type = "textbook">
<title>
<price>
<author>
E3'

【書類名】　　要約書

【要約】

【課題】複数のウェブサイトの情報を１つのウェブ文書上に合成することが容易にしかも汎用的に行える文書合成方法および文書合成装置を提供する。

【解決手段】少なくとも、インターネットにおけるWWW上のマークアップ言語で記述された第１の文書のインターネット上の所在と、第１の文書から抽出する部分文書の範囲と、合成用の第２の文書上の前記部分文書の挿入位置と、前記挿入位置に挿入される前記部分文書を含む前記第２の文書上の文書構造を変換すべき範囲と、前記文書構造を所望の文書構造に変換するための変換ルールを記述したファイルの識別情報とをマークアップ言語により記述した第２の文書に従って、前記第１の文書から前記部分文書を抽出して、その部分文書を前記第２の文書上の前記指定された合成位置に挿入するとともに、前記変換ルールを用いて前記第２の文書上の前記指定された範囲の文書構造を変換する。

【選択図】　　図２

出　願　人　履　歴　情　報

識別番号　　〔０００００３０７８〕

１．変更年月日　　１９９０年　８月２２日
［変更理由］　新規登録
住　所　神奈川県川崎市幸区堀川町７２番地
氏　名　株式会社東芝

２．変更年月日　　２００１年　７月　２日
［変更理由］　住所変更
住　所　東京都港区芝浦一丁目１番１号
氏　名　株式会社東芝